https://doi.org/10.31891/2307-5732-2026-365-67
УДК 004.65:004.75

BORODII IVAN
Ternopil Ivan Puluj National Technical University
https://orcid.org/0009-0005-4986-4429
e-mail: ivanborodii@tntu.edu.ua
OSUKHIVSKA HALYNA
Ternopil Ivan Puluj National Technical University
https://orcid.org/0000-0003-0132-1378
e-mail: osukhivska@tntu.edu.ua

COMPARATIVE ANALYSIS OF IOT DATA PROCESSING ARCHITECTURES FOR ENVIRONMENTAL MONITORING ON RASPBERRY PI 5

The paper represents a comparative study of the performance of hybrid micro-batch processing strategies for environmental data using the Raspberry Pi 5 microcomputer. The experiment is based on a real stream of data from BME688 and SCD41 sensors, collected over 142 hours. The study compares six data storage systems of different architectural classes: SQLite, PostgreSQL, DuckDB, Polars, ClickHouse, and Delta Lake. The experiment involved micro-batch data processing with the accumulation of 50-record batches and the recording of system performance metrics. Statistical parameters of recording time, processor load, and RAM usage, including mean values, standard deviation, and quantile latency, as well as the amount of disk space required to store data, were used to analyse the results. The obtained experimental measurements allow a direct comparison of the behaviour of different data storage architectures under identical operating conditions on a resource-constrained edge device. The analysis focuses not only on average execution performance but also on the variability of delays that may occur during data ingestion operations. Such an approach makes it possible to evaluate the stability and reliability of each system in real-time data processing scenarios. The results show significant performance differences between the systems studied. The lowest delays in recording micro-batches were recorded for SQLite, ClickHouse, and Polars, while the use of Delta Lake is accompanied by significant overhead costs of computing resources on peripheral equipment. Additional observations indicate that lightweight or embedded data processing solutions demonstrate better suitability for edge computing environments compared to complex distributed architectures that require additional processing layers. The study provides an empirical basis for choosing optimal data storage architectures in real-time environmental monitoring systems running on devices with limited hardware resources and contributes to the development of efficient IoT data processing pipelines for edge-based monitoring infrastructures.

Keywords: *Internet of Things, environmental monitoring, Edge Computing, micro-batch data processing, Raspberry Pi.*

БОРОДІЙ ІВАН, ОСУХІВСЬКА ГАЛИНА
Тернопільський національний технічний університет імені Івана Пулюя

ПОРІВНЯЛЬНИЙ АНАЛІЗ АРХІТЕКТУР ОБРОБКИ IOT-ДАНИХ ДЛЯ ЕКОЛОГІЧНОГО МОНІТОРИНГУ НА RASPBERRY PI 5

У роботі представлено порівняльне дослідження продуктивності гібридних стратегій мікропакетної обробки екологічних даних з використанням мікрокомп'ютера Raspberry Pi 5. Експеримент базується на реальному потоці даних з датчиків BME688 і SCD41, зібраних протягом 142 годин. У дослідженні порівнюються шість систем зберігання даних різних архітектурних класів: SQLite, PostgreSQL, DuckDB, Polars, ClickHouse і Delta Lake. Експеримент передбачав обробку мікропакетів даних із накопиченням пакетів із 50 записами та фіксацією показників продуктивності системи. Для аналізу результатів використовувалися статистичні параметри часу запису, завантаження процесора та використання оперативної пам'яті, включаючи середні значення, стандартне відхилення та показники квантильної затримки, а також обсяг дискового простору, необхідного для зберігання даних. Отримані експериментальні вимірювання дозволяють безпосередньо порівняти поведінку різних архітектур зберігання даних в однакових умовах експлуатації на периферійному пристрої з обмеженими ресурсами. Аналіз зосереджується не тільки на середній продуктивності виконання, але й на мінливості затримок, які можуть виникати під час операцій збору даних. Такий підхід дозволяє оцінити стабільність і надійність кожної системи в сценаріях обробки даних у реальному часі. Результати показують значні відмінності в продуктивності між досліджуваними системами. Найменші затримки в записі мікропакетів були зафіксовані для SQLite, ClickHouse і Polars, тоді як використання Delta Lake супроводжується значними накладними витратами обчислювальних ресурсів на периферійному обладнанні. Додаткові спостереження показують, що легкі або вбудовані рішення для обробки даних демонструють кращу придатність для периферійних обчислювальних середовищ у порівнянні зі складними розподіленими архітектурами, які вимагають додаткових рівнів обробки. Дослідження надає емпіричну основу для вибору оптимальних архітектур зберігання даних у системах моніторингу навколишнього середовища в режимі реального часу, що працюють на пристроях з обмеженими апаратними ресурсами, та сприяє розробці ефективних конвеєрів обробки даних IoT для периферійних інфраструктур моніторингу.

Ключові слова: *Інтернет речей, екологічний моніторинг, периферійні обчислення, мікро-пакетна обробка даних, Raspberry Pi.*

Стаття надійшла до редакції / Received 11.02.2026
Прийнята до друку / Accepted 11.03.2026
Опубліковано / Published 28.05.2026

Problem statement

In the modern world, monitoring the environment is an important task for ensuring ecosystem security and supporting sustainable development. The widespread adoption of Internet of Things (IoT) technologies creates new opportunities for the continuous collection of environmental indicators in real time. The use of sensor networks provides detailed time series of data on gas concentrations, temperature, humidity, and other microclimate parameters, creating a basis for further analysis of atmospheric conditions.

Traditional IoT data processing architectures typically involve transmitting raw measurements to remote cloud

or centralized data storage systems. However, transmitting large volumes of telemetry data directly to cloud storage can result in significant delays, additional data transmission costs, and dependence on the stability of the network infrastructure. In this regard, the concept of Edge Computing is becoming increasingly widespread, which involves performing part of the computational operations directly on edge nodes located near data sources.

The use of modern microcomputers, such as the Raspberry Pi, enables the deployment of systems for collecting and preprocessing environmental data directly at the edge. Such devices can perform functions of local storage, aggregation, and initial processing of data before its further transmission to centralized analytical systems. At the same time, the use of edge devices is accompanied by limitations on computational resources, specifically RAM capacity, processor performance, and I/O speed.

Under these conditions, a key challenge is choosing an architecture for data storage and processing that ensures minimal write latency, stable utilization of computational resources, and efficient use of disk space. In modern information systems, different approaches can be used for this purpose, including traditional transactional databases, analytical columnar systems, and new architectures such as the Data Lakehouse. Each of these approaches has its own features in terms of data organization, processing mechanisms, and the use of system resources.

Despite a significant amount of research in the field of data management systems, the issue of the effective use of different data storage formats in scenarios involving micro-batch data processing of IoT data on edge devices remains under-researched. It is particularly important to conduct experimental studies under real-world operating conditions of environmental monitoring systems, which makes it possible to evaluate the performance of various technological solutions on devices with limited computational resources.

Analysis of recent studies and publications

The research paper [1] presents a systematic review and comparative evaluation of modern SQL-on-Hadoop and Lakehouse systems. Benchmark tests were conducted using a Raspberry Pi 4 Model B microcomputer. The impact of vectorized execution and JIT compilation on the speed of analytical queries was investigated. The results confirmed that vectorization is a determining factor in performance for complex queries even on peripheral devices.

Paper [2] is devoted to an empirical study of the execution speed of analytical queries in PostgreSQL, SQLite, Pandas, DuckDB, and Polars systems on datasets up to 100 GB. A critical difference in data loading latency was found between traditional relational databases and “zero-copy” columnar mechanisms. The conclusions highlight the advantages of DuckDB and Polars for local data processing, which is highly relevant for Edge Computing.

Article [3] proposes SciTS, a tool for comparative analysis of time-series databases, developed specifically for use in scientific research and the industrial IoT sector. The results of this study indicate that QuestDB demonstrates the best data ingestion speed, while TimescaleDB proves to be more efficient for complex analytical operations.

Study [4] describes an intelligent air quality monitoring system in an urban environment based on IoT technologies and machine learning algorithms, where the use of a Raspberry Pi 4 is considered as the main component for collecting readings from gas sensors and their subsequent transmission. The results showed a prediction accuracy of over 90%, confirming the effectiveness of using intelligent models for real-time environmental analysis.

Work [5] is dedicated to evaluating the performance and power consumption of containerized solutions on resource-constrained IoT edge gateways. The authors investigated the impact of virtualization on CPU and memory usage and power consumption during the execution of typical IoT tasks and demonstrated the feasibility of using such technologies on microcomputers to create stable distributed data collection networks.

In the study [6], the characteristics of the BME688 gas sensor were evaluated under controlled conditions simulating an open environment. The results confirm the suitability of the BME688 for use in high-precision edge systems for environmental monitoring.

A scientific paper [7] investigated the efficiency of data loading and storage in Data Lakehouse architectures. The authors conducted a comparative analysis of loading speed and storage stability for data volumes up to 7 GB using Apache Spark. The results demonstrated that Delta Lake provides the highest data loading speed, while Iceberg is the most effective for minimizing disk space usage.

Study [8] is dedicated to a comparative analysis of the performance of Continuous Processing and Micro-batch modes in Spark Structured Streaming. The authors found that continuous processing provides ultra-low latency of up to 2 ms when using a Rate source, but micro-batch data processing remains more stable when working with high-intensity sources such as Kafka. This allows for the optimization of IoT stream data processing pipelines.

In paper [9], a two-level architecture is proposed for scheduling hybrid workflows in cloud-edge systems. The proposed approach is based on a resource estimation algorithm using gradient descent (GDS) and a cluster-based task scheduling technique (C-HWPS). An important component of the model is the consideration of heterogeneous edge nodes, particularly Raspberry Pi devices, for performing data preprocessing near IoT sources.

Research [10] conducted a comparative study of Apache Spark’s performance when executing full ETL processes in Java, Python, and Scala using the Apache Iceberg format. Experiments showed that Python demonstrates an advantage when working with small datasets due to its high-level APIs and low JVM overhead.

Purpose of the article

The aim of the work is to evaluate the efficiency of using different data storage architectures for implementing a hybrid strategy for micro-batch data processing of IoT environmental monitoring data on the Raspberry Pi 5. To achieve this goal, the work conducts a comparative evaluation of the performance of several architectural approaches to data storage organization, specifically transactional, analytical, and Data Lakehouse solutions, using an implemented

experimental system.

The primary focus is on analyzing the performance of micro-batch data writes, as well as evaluating the use of the peripheral device's computational resources during data loading operations. To this end, key system metrics are recorded, including the execution time of write operations, CPU load, the amount of RAM used, and the physical memory capacity. This enables an objective comparison of different data management solutions and helps identify the most effective approaches to organizing information in environmental monitoring platforms operating on Edge Computing nodes.

Presentation of the main material

A comparative analysis of IoT data processing architectures for environmental monitoring is conducted for six different data storage systems chosen for testing on an edge node. Specifically, the relational DBMSs PostgreSQL and SQLite, the analytical systems ClickHouse and DuckDB, as well as the Polars library using the Parquet format and the Data Lakehouse technology based on the Delta Lake format using the PySpark library were selected. All data storage systems under investigation were deployed locally.

The experimental study was conducted on a Raspberry Pi 5 microcomputer with a quad-core Broadcom BCM2712 processor and 8 GB of LPDDR4X-4267 RAM, running under the Linux kernel version 6.12.62. The project configuration is deployed on a 128 GB MicroSD card and serves as the primary storage for all data storage systems. The use of the Raspberry Pi 5 ensures stable operation during intensive I/O operations and provides sufficient computing power for conducting the comparative experiment.

The system's software implementation was developed using the Python 3.13.5 programming language. The implementation of each data system relies on optimized libraries that enable the measurement of data batch loading performance. In particular, the Apache PySpark framework and the delta-spark library are used to work with Delta Lake, which require initialization of a Java Virtual Machine (JVM) to ensure transactional integrity and support schema evolution, a characteristic feature of Data Lakehouse systems. Relational interaction with PostgreSQL is implemented via the psycopg2 library using the executemany batch data loading function. Analytical processing in ClickHouse is performed using the official clickhouse-driver with the MergeTree engine configured. The built-in SQLite database uses the standard sqlite3 module in transaction mode. Data loading into DuckDB is achieved by using DataFrame objects via the Apache Arrow format, and the Polars library provides autonomous data storage in the Parquet columnar format. The features of the software implementation of data storage systems are listed in Table 1.

Table 1

Features of the software implementation of data storage systems

Data storage system	Python library	Data loading method	Features of software implementation
Delta Lake	pyspark (version 4.1.1), delta-spark (4.1.0)	df.write.save()	Requires JVM and Spark session initialization. Uses Java 17.0.10
PostgreSQL	psycopg2-binary (2.9.11)	cur.executemany()	Loading data packages with automatic table creation
ClickHouse	clickhouse-driver (0.2.10)	client.execute()	Using MergeTree engine
SQLite	sqlite3 (built-in)	conn.executemany()	Embedded database
DuckDB	duckdb (1.4.4)	INSERT SELECT FROM df	Direct data loading from Polars DataFrame
Polars	Polars (1.38.1)	df.write_parquet()	Autonomous recording of data packages in Parquet format

Raw environmental data is collected using the SCD41 photoacoustic sensor and the BME688 smart gas module, which are connected to the Raspberry Pi via the I2C interface. Interaction with the BME688 is implemented via the bme680 library, which measures temperature, humidity, pressure, and the resistance of the gas sensor. The SCD41 driver is built on the smbus2 library and provides low-level transmission of 16-bit commands, including a bus stabilization procedure and the initiation of periodic readings. Data received from the BME688 and SCD41 sensors is loaded into the bme and scd tables, respectively.

The input set of environmental parameters is generated based on data from two modules: the BME688 smart gas sensor provides values for temperature, humidity, atmospheric pressure, and gas module resistance, while the SCD41 photoacoustic sensor measures CO2 concentration, as well as duplicate readings for temperature and humidity. Each generated record is accompanied by a precise timestamp of the measurement, which ensures the integrity of the time series and enables an objective comparison of the processing speed of identical data across all storage systems being studied.

The monitoring process is organized according to the principle of micro-batch processing: values are read every 90 seconds and are mandatorily saved in a temporary JSON buffer file to prevent data loss. When a package of 50 records is accumulated, a comparative testing procedure is initiated, which sequentially runs all six data loading functions for each system. Selecting a batch size of 50 records is optimal for balancing data update speed and the performance of the Raspberry Pi 5's disk, ensuring efficient storage filling without overloading the system with service information.

The performance measurement methodology is based on recording system metrics in real time using the psutil

library. Three key metrics are used for quantitative analysis to compare the resource consumption of each system under test: execution time, CPU load, and RAM usage. Each database is initialized via isolated context managers that close the connection immediately after the write operation is complete. The write time to Delta Lake is measured without accounting for the initialization of the Spark session, which gives objective data on the format's performance. The results are automatically stored in the system_metrics analytical table of the DuckDB database. The structural description of the system_metrics data is shown in Table 2.

Table 2

Structure of system_metrics table

Metric Name	Unit of measurement	Method of obtaining	Description
duration_sec	sec	time.perf_counter()	The time taken to load the data package.
cpu_pct	%	psutil.cpu_percent()	CPU load. The cpu_percent method returns the average load across all cores.
ram_mb	MB	psutil.Process().memory_info()	The amount of RAM consumed by the Python script itself while writing data.

The system architecture is based on a clear separation of functions, where the logic for interacting with hardware modules is completely isolated from analytical processes. This makes it easy to scale the solution and add new data storage methods without modifying the core data collection code. To ensure the resilience of the experiment, an automatic recovery strategy and multi-level logging have been implemented. The stability of the sequence of operations is ensured by the systemd service, which initiates a restart of the orchestrator module in the event of a critical failure, guaranteeing the continuity of recording environmental data and system metrics via the sensors.service file. The distribution of tasks among the project's main software modules is shown in Table 3.

Table 3

Structure of the system's software implementation

Software module	Description
main.py	Coordinating sensor operations; initiating batch processing and writing to a table
generic.py	Implementation of data loading methods; logging of system metrics; initialization of constant variables used by other modules
bme.py / scd.py	Reading sensor data via the I2C interface
sensors.service	Ensuring the project starts automatically when the OS boots and monitoring the continuous execution of processes

The proposed experimental design ensures identical conditions for each storage system, enabling the most objective comparison results. This provides a solid scientific and technical foundation for identifying the most effective data processing methods on low-power devices in real-time conditions. A general block diagram of the system for evaluating the effectiveness of hybrid micro-batch data processing strategies for IoT data in environmental monitoring systems is shown in Figure 1.

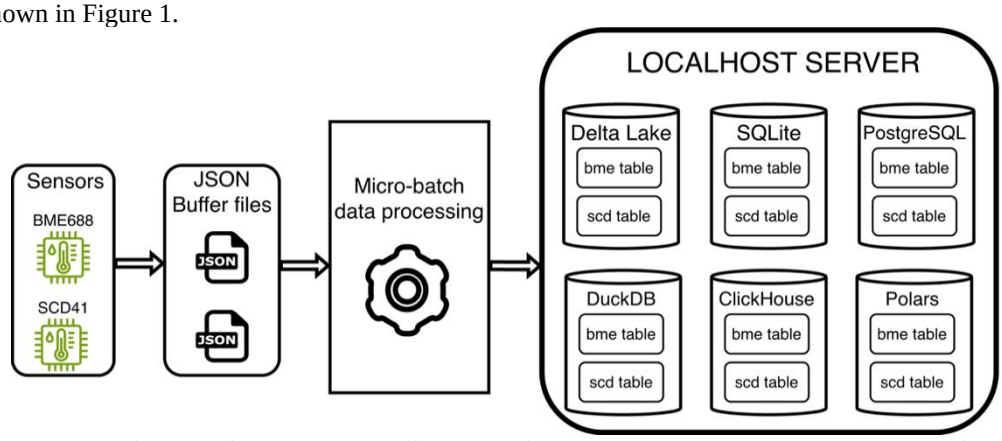

Fig. 1. Block diagram of a system for evaluating the effectiveness of hybrid micro-batch data processing strategies for IoT data in environmental monitoring systems

During the 142-hour measurement period, 250 data micro-batches were received and processed from the BME688 and SCD41 sensors. Each micro-batch contained 50 records of environmental parameters, which were sequentially uploaded to the six data storage systems under study.

To evaluate performance, statistical analysis was performed on the measurement results for recording time, CPU load, and RAM usage. In addition to the mean values, the standard deviation, as well as the 95th and 99th quantiles of the distribution of the corresponding indicators, were determined. This approach makes it possible to reflect not only the

average performance of the systems but also the features of the distribution of delays in recording operations. A separate aspect of the study is the assessment of the disk space occupied by the accumulated data in the bme and scd tables in each system. This enables a comparison of the efficiency of different methods of data organization on the peripheral device when working with identical arrays of environmental indicators. Table 4 presents statistical indicators of time, CPU load, and RAM usage during the execution of micro-batch data write operations, as well as the physical size of the data after the completion of a full measurement cycle for each data storage system.

Table 4

Performance metrics for data storage architectures in environmental monitoring systems

Architectures / Metrics	Delta Lake	PostgreSQL	ClickHouse	SQLite	DuckDB	Polars
Statistical metrics for micro-batch data write times						
Average write time (sec)	2,58	0,08	0,02	0,02	0,13	0,02
Standard deviation	5,34	0,18	0,04	0,02	0,33	0,06
95th percentile	10,91	0,22	0,03	0,06	0,36	0,18
99th percentile	30,23	0,76	0,23	0,07	0,73	0,3
Statistical metrics for CPU load						
Average load, %	82,44	48,74	46,51	31,24	36,12	32,91
Standard deviation	12,31	33,42	35,1	40,54	37,02	43,67
95th percentile	99,8	100	100	100	100	100
99th quantile	100	100	100	100	100	100
Statistical indicators of RAM usage						
Average usage, MB	94,39	96,97	97,31	97,39	98,07	100,8
Standard deviation	16,73	14,02	13,61	13,58	14,22	15,45
95th quantile	118,61	120,61	121,66	120,8	123,12	129,48
99th quantile	147	148,14	146,89	149,45	165,44	177,48
Physical size of data stores: Disk space of bme and scd tables						
Disk space, MB	12,9	0,64	0,238	0,692	1,6	0,024

The measurements obtained are based on entries in the system_metrics analytical table, which recorded each individual stage of data package loading. The data presented enables a direct quantitative comparison of different architectural solutions under identical hardware operating conditions, without accounting for the time required to initialize the software environments. A comparative diagram of disk space usage and average write time for the data systems under study is shown in Figure 2.

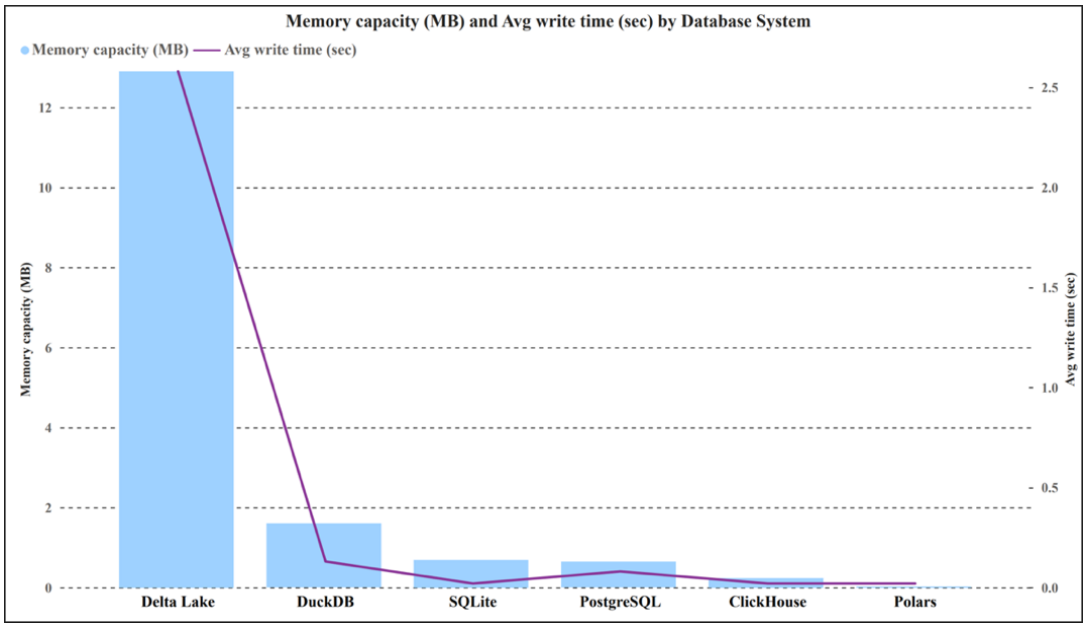

Fig. 2. Comparative chart of disk space usage and average write time for the data systems under study

Conclusions

This article presents a comparative analysis of six data storage system architectures when implementing a hybrid strategy for micro-batch processing of IoT data on a Raspberry Pi 5 Edge Computing node. Specifically, the study utilized SQLite, PostgreSQL, DuckDB, Polars, ClickHouse, and Delta Lake.

Key system metrics were analyzed, including operation execution time, CPU load, and RAM usage. Statistical analysis of the results was performed based on mean values, standard deviation, and the extreme values of the distribution of operation execution delays.

The shortest average micro-batch data write time with the shortest average operation execution time was recorded for SQLite, ClickHouse, and Polars, at approximately 0.02 seconds. PostgreSQL and DuckDB exhibit slightly longer write times, which is attributed to the characteristics of their internal transaction management and data processing mechanisms.

The longest operation execution delays are observed in the Delta Lake system, where the average micro-batch data write time is 2.58 seconds. This behavior is explained by the complexity of the Data Lakehouse architecture and the need to use the Apache Spark environment, maintain transaction logs, and support metadata.

An analysis of hardware resource utilization revealed that the highest CPU load was observed in Delta Lake, where the average CPU utilization exceeded 82%. For the other systems, this metric ranged from 31% to 49%, indicating significantly lower overhead in computational resources.

Additional analysis of disk space usage revealed significant differences among the systems under study. The largest storage volume was recorded for Delta Lake (12.9 MB), while the smallest data sizes were observed in Polars (0.024 MB) and ClickHouse (0.238 MB). This difference is due to the use of columnar storage formats and efficient data compression algorithms.

The results indicate that for tasks involving the real-time collection and processing of IoT data for environmental monitoring on edge devices, lightweight or columnar data storage systems—specifically SQLite, ClickHouse, and Polars—are the most effective. The use of full-fledged Lakehouse solutions on low-power edge devices may be accompanied by significant overhead in computational resources and increased delays in write operations.

Further research should focus on integrating methods for intelligent analysis of environmental data directly on the edge node using compact machine learning models to detect abnormal atmospheric conditions.

References

1. Szoke M. A. Modern alternatives to hive: a systematic review and single-node benchmark of SQL-on-Hadoop and lakehouse engines [Electronic resource] / M. A. Szoke // Information Systems. – 2026. – Vol. 136. – 102635. – Mode of access: https://doi.org/10.1016/j.is.2025.102635 (date of access: 10.03.2026). – Title from screen.

2. Hasan J. Benchmarking analytical query performance: an empirical study of query optimization in modern data systems [Electronic resource] / J. Hasan. – Technical Report. – 2025. – Mode of access: https://arxiv.org (date of access: 10.03.2026). – Title from screen.

3. Mostafa J. SciTS: a benchmark for time-series databases in scientific experiments and industrial Internet of Things [Electronic resource] / J. Mostafa, S. Chilingaryan, S. Wehbi, A. Kopmann // Proceedings of the 34th International Conference on Scientific and Statistical Database Management (SSDBM 2022). – 2022. – P. 1–12. – Mode of access: https://doi.org/10.1145/3538712.3538723 (date of access: 10.03.2026). – Title from screen.

4. Alsamrai O. Real-time intelligent monitoring of outdoor air quality in an urban environment using IoT and machine learning algorithms [Electronic resource] / O. Alsamrai, M. D. Redel-Macias, M. P. Dorado // Applied Sciences. – 2025. – Vol. 15. – 9088. – Mode of access: https://doi.org/10.3390/app15169088 (date of access: 10.03.2026). – Title from screen.

5. Raza S. M. Empirical performance and energy consumption evaluation of container solutions on resource constrained IoT gateways [Electronic resource] / S. M. Raza, J. Jeong, M. Kim, B. Kang, H. Choo // Sensors. – 2021. – Vol. 21. – 1378. – Mode of access: https://doi.org/10.3390/s21041378 (date of access: 10.03.2026). – Title from screen.

6. Panzardi E. Assessing BME688 sensor performance under controlled outdoor-like environmental conditions [Electronic resource] / E. Panzardi, A. Fort, V. Vignoli, I. Cappelli, L. Gaioni, M. Verzeroli, S. Dello Iacono, A. Flammini // Sensors. – 2025. – Vol. 25. – 7102. – Mode of access: https://doi.org/10.3390/s25237102 (date of access: 10.03.2026). – Title from screen.

7. Borodii I. Research on the efficiency of data loading and storage in Data Lakehouse architectures for the formation of analytical data systems [Electronic resource] / I. Borodii, H. Osukhivska // Information Technology: Computer Science, Software Engineering and Cyber Security. – 2025. – Vol. 4. – P. 28–36. – Mode of access: https://doi.org/10.32782/1T/2025-4-4 (date of access: 10.03.2026). – Title from screen.

8. Fedorovych I. Performance benchmarking of continuous processing and micro-batch modes in Spark Structured Streaming [Electronic resource] / I. Fedorovych, H. Osukhivska, N. Lutsyk // Proceedings of the 4th International Workshop on Information Technologies: Theoretical and Applied Problems (ITTAP 2024). – 2024. – P. 80–90. – Mode of access: https://ceur-ws.org/Vol-3896/paper5.pdf (date of access: 10.03.2026). – Title from screen.

9. Alsurdeh R. Hybrid workflow scheduling on edge cloud computing systems [Electronic resource] / R. Alsurdeh, R. N. Calheiros, K. M. Matawie, B. Javadi // IEEE Access. – 2021. – Vol. 9. – P. 134783–134799. – Mode of access: https://doi.org/10.1109/ACCESS.2021.3116716 (date of access: 10.03.2026). – Title from screen.

10. Borodii I. Comparative analysis of large data processing in Apache Spark using Java, Python and Scala [Electronic resource] / I. Borodii, I. Fedorovych, H. Osukhivska, D. Velychko, R. Butsii // Proceedings of the 3rd International Workshop on Computer Information Technologies in Industry 4.0 (CITI 2025), Ternopil, Ukraine, 11–12 June 2025. – CEUR Workshop Proceedings. – Vol. 4057. – P. 189–198. – Mode of access: https://ceur-ws.org/Vol-4057/paper13.pdf (date of access: 10.03.2026). – Title from screen.