

A. Kapiton¹, T. Franchuk², D. Tyshchenko², R. Zakharov²

¹National University «Yuri Kondratyuk Poltava Polytechnic», Poltava, Ukraine

²State University of Trade and Economics, Kyiv, Ukraine

DEVELOPMENT OF DECENTRALIZED FORECASTING SYSTEMS BASED ON MULTI-CLOUD INFRASTRUCTURE

Abstract. Relevance. The study substantiates the need for a cloud environment that ensures a high level of system scalability, enabling the integration of new production facilities, workshops, and equipment without the need for capital investment in proprietary server infrastructure. **The subject of the study** is the processes of developing and operating decentralized forecasting systems based on multi-cloud infrastructure. **The aim of the study** is to analyze modern methods for developing decentralized forecasting systems based on multi-cloud infrastructure and to enhance their efficiency, reliability, and fault tolerance. **The research objectives** are to investigate the architectural aspects of creating decentralized forecasting systems in a cloud environment; to examine the effectiveness of using distributed computing for technical diagnostics; and to determine a software solution for time-series forecasting using recurrent neural networks. to analyze cloud platforms and machine learning libraries for distributed time-series modeling; to identify a key challenge regarding the end-to-end analysis of various neural network architectures and formulate technical requirements to overcome this limitation.

Research methods. The methodology encompasses systems analysis, mathematical modeling, blockchain technologies, and machine learning methods. **Research results.** The feasibility of employing cloud and distributed architectures to address diagnostic tasks is substantiated, driven by the specific requirements for processing large volumes of data continuously generated by primary sensors. It is determined that a necessary prerequisite is the development and investigation of a time-series forecasting software system based neural network models utilizing cloud computing; such a system must implement the complete data processing cycle from acquisition to forecast generation and quality assessment thereby enabling its subsequent application in the field of computer engineering. A structured plan for implementing the system's end-to-end operational cycle is presented. **Conclusions.** The implementation and deployment of such forecasting systems within a cloud infrastructure, utilizing containerization and orchestration via managed services align with modern engineering standards that ensure high fault tolerance and horizontal scalability for analytical platforms.

Keywords: forecasting systems; cloud infrastructure; distributed systems; data synchronization; network latency; data orchestration; machine learning; computer engineering; information technology.

1. Introduction

1.1 Relevance of the topic

A cloud-based, distributed time-series forecasting system for vibration diagnostics is essential for preventing industrial accidents and transitioning from scheduled repairs to condition-based maintenance. It analyzes the oscillations and vibrations of machine tools, turbines, or engines to detect hidden defects well before the equipment fails. The system's primary purpose is to predict equipment failures (Predictive Maintenance) and calculate remaining useful life (RUL) for precise maintenance planning. Continuous, automated real-time sensor monitoring eliminates the need for manual measurements.

The system ensures early defect detection by identifying micro-cracks, imbalance, bearing wear, and shaft misalignment. This approach minimizes financial losses by eliminating the risk of sudden conveyor shutdowns and completely preventing prolonged industrial production downtime.

1.2 Analysis of literature sources and problem definition

A thorough exploration of the topic requires examining methodological approaches to evaluating decentralized forecasting systems in a cloud environment. Since this area has been overlooked in recent publications, the design of such systems currently represents one of the most pressing directions in contemporary research.

C. Chatfield addresses general issues of time series forecasting [1].

D. Popovic investigates computational intelligence in time series forecasting [2].

G. James, D. Witten, T. Hastie, and R. Tibshirani examine the theoretical and practical aspects of statistical learning [3].

In other the works are explore the applied aspects of encoder-decoder architectures based on recurrent neural networks [4].

S. Amari discusses the backpropagation and stochastic gradient descent methods [5].

Researchers are specialists in wavelet analysis and financial time series forecasting [6, 7].

The authors describe the characteristics and issues associated with the problems of investigate applied aspects, specifically forecasting on the NSE stock market using deep learning models [8].

Other articles open the topic of examine the learning stages involved in recognizing long-term dependencies using gradient descent, noting it as a complex task [9].

A number of works are devoted to the problems to analyze this issue through the lens of Long Short-Term Memory and investigate and empirically evaluate recurrent neural networks with gating mechanisms in sequence modeling tasks, and also conduct a comparative analysis of financial time series forecasting [10–13].

A number of researchers examine modern development trends and architectural features of

computers, as well as the fundamentals of modeling decentralized forecasting systems based on cloud platforms [14–16]. An objective study of the problem requires an analysis of existing solutions in the field of cloud-based distributed forecasting systems. Previous studies overlooked this aspect, making the development methodology for such decentralized platforms a central topic of current scientific discussions.

1.3 Aim and objectives of the study

The aim of the study is to analyze methods for developing decentralized forecasting modern systems, which based on multi-cloud infrastructure, enhance their efficiency, reliability, fault tolerance.

To achieve the stated objective, the following scientific and practical tasks have been formulated:

- investigating architectural solutions for constructing decentralized forecasting systems in a cloud environment;
- evaluating the effectiveness of using distributed computing in technical diagnostics processes based on the continuous analysis of large datasets from primary sensors;
- identify a software solution for time-series forecasting using recurrent neural networks with GRU and LSTM architectures;
- analyze cloud platforms and machine learning libraries for distributed time-series modeling;
- identify a key challenge for the end-to-end analysis of various neural network architectures and formulate technical requirements to address this limitation.

2. Research materials and methods

The object of the study is the process of forecasting distributed time series (large volumes of diagnostic data) in decentralized multi-cloud infrastructures. Research hypothesis: The implementation and deployment of forecasting systems within a cloud infrastructure, utilizing Docker containerization and orchestration via managed services, enable high fault tolerance and horizontal scalability for analytical platforms in accordance with modern engineering standards.

The study formalizes the stages of the development lifecycle for predictive systems at the intersection of Data Science methodology, cloud engineering, and the diagnostic domain, enabling the formulation of a structured architecture for an end-to-end operational cycle. A comparative analysis of cloud technologies relevant to forecasting distributed time series in computer engineering was conducted. Neural network modeling tools based on the TensorFlow, PyTorch, and Prophet libraries were investigated.

A critical scientific and practical barrier was identified: the lack of an integrated environment with a graphical user interface for the comparative analysis of heterogeneous architectures within a unified computational pipeline. Based on the identified limitations, a set of technical requirements was established for the proposed software system, designed to minimize these shortcomings and ensure the practical value of the results obtained.

3. Results of development and research

3.1 Design and practical implementation of decentralized intelligent forecasting systems

The design and practical implementation of decentralized intelligent forecasting systems based on heterogeneous multi-cloud infrastructure entail a complex set of critical architectural and technological challenges. Key among these are the systemic complexity of orchestrating distributed data arrays, heightened risks regarding the confidentiality and integrity of the information perimeter, and deterministic latencies in inter-node data exchange. The primary factors negatively affecting the efficiency of such systems are the processes of ensuring real-time data consistency and synchronization, the structural and functional heterogeneity of the cloud platforms employed, and the high computational and network overhead associated with maintaining communication between distributed components. The need to employ cloud and distributed architectures for vibration diagnostics tasks stems from the specific requirements of processing large datasets (Big Data) continuously generated by primary sensors. Since modern vibration sensors produce high-frequency signals - yielding thousands of measurements per second - parallel processing of these data streams is implemented using distributed computing frameworks. Utilizing a cloud environment ensures a high level of system scalability, enabling the integration of new production facilities, workshops, and machinery without the need for capital investment in on-premises server infrastructure. Furthermore, mathematical modeling and the training of predictive AI neural network structures - specifically recurrent network and transformer architectures - require substantial computational resources, which are efficiently provided by cloud-based graphics processing units (GPUs).

Above all, this architecture implements the concept of data centralization, aggregating diagnostic information from all the enterprise's remote sites into a single, consolidated monitoring interface to facilitate management decision-making [1, 4, 8].

Time series forecasting is a fundamental applied task that is critical to the financial, energy, telecommunications, medical, and computer science sectors. Traditional statistical approaches - such as ARIMA and exponential smoothing - demonstrate satisfactory performance only when dealing with stationary data and stable variance. However, their predictive capability is significantly limited in the presence of heteroscedasticity, or structural shifts. To adequately model this processes in such series, modern researchers employ networks.

In particular, Hochreiter and Schmidhuber's LSTM architecture overcame tasks, enabling some effective detection of long-range dependencies in data sequences. A simplified variant, the Gated Recurrent Unit, used fewer parameters while maintaining comparable accuracy. A comparative study by Siami-Namini et al. confirmed that LSTM and GRU reduce RMSE by an average of 84–87% compared to ARIMA when applied to financial time series [1, 5, 6, 13].

Despite the powerful toolsets for building neural network models available in libraries such as TensorFlow, PyTorch, and Prophet, they lack an integrated environment with a graphical user interface for the comparative analysis of architectures within a unified computational pipeline. Consequently, the development of a custom software suite offering these capabilities constitutes a practical contribution of this research. Implementing and deploying such predictive systems within a cloud infrastructure leveraging Docker containerization and orchestration via managed services (such as AWS Fargate, Google Cloud Run, or Azure Container Apps) reflects modern engineering standards that ensure high fault tolerance and horizontal scalability for analytical platforms.

3.2 A structured plan for implementing the end-to-end system work cycle

A necessary requirement is to develop and investigate a time-series forecasting software system based on GRU and LSTM neural network models using cloud computing; this system must implement the complete data processing cycle - from data loading to generating a forecast and evaluating its quality - for subsequent application in computer engineering. The development and research of such a system are broken down into distinct stages that combine Data Science methodology, cloud engineering, and the specifics of vibration diagnostics.

A structured plan for implementing the end-to-end system work cycle is presented in Fig.1.

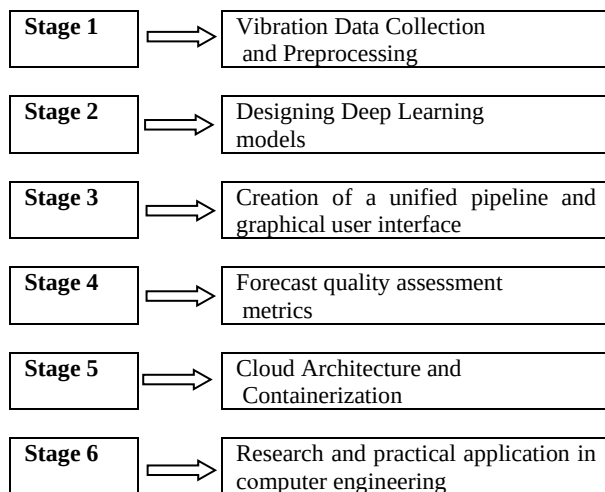


Fig. 1. A structured plan for implementing the end-to-end system work cycle

Stage 1. Vibration Data Collection and Preprocessing (Data Engineering). Vibration signals are high-frequency time series (typically ranging from 2 to 50 kHz). The full cycle begins with data ingestion and adaptation. Loading: Organizing streaming or batch data ingestion via APIs (e.g., from industrial IoT gateways such as those from SKF or Siemens). Cleaning and Resampling: Removing anomalies, filling in missing values, downsampling (if long-term trend analysis is required), or applying sliding windows to generate data samples. Feature Engineering:

The effectiveness of GRU/LSTM architectures depends directly on the amplitude-frequency properties of the data. Temporal parameters (RMS, crest factor, kurtosis) and spectral features calculated via FFT or DWT are evaluated. Normalization: Scaling data to the $[0, 1]$ or $[-1, 1]$ range using MinMaxScaler or StandardScaler (critical for the convergence of recurrent networks) [1, 6-14].

Stage 2. Designing Deep Learning models. Given the time-dependent nature of vibration processes, where the current state of structural elements (such as rotor shafts or rolling-element bearings) is directly determined by their prior history.

In particular, the Long Short-Term Memory (LSTM) architecture utilizing a mechanism of such functional gates enables the effective approximation of long-term nonlinear wear trends and failure prediction. As a more computationally optimized alternative, the Gated Recurrent Unit (GRU) modification is employed; it implements a simplified structure based on two gates that control data flow.

It is computationally "lighter," trains faster in the cloud, and often demonstrates similar or superior accuracy on short time series.

Hybrid approaches (optional): A combination of 1D-CNN (for automatic feature extraction from vibration signals) with subsequent LSTM/GRU layers for trend forecasting [1, 8, 14].

Stage 3. Creation of a unified pipeline and graphical user interface. To overcome the limitations of standard libraries, a monolithic or microservice-based application is being developed (e.g., using Python/FastAPI/Streamlit or Dash): Configuration module: Via the GUI, the user selects the model (LSTM or GRU) and specifies hyperparameters (number of layers, batch size, epochs, forecasting window).

Unified pipeline: Single-button process execution: Data loading → Preprocessing → Training/Validation → Result visualization [1, 4, 8].

Stage 4. Forecast quality assessment metrics: For comparative analysis, the interface includes a mathematical module that evaluates the deviation of the predicted vibration from the actual values: MAE (Mean Absolute Error): Indicates the average absolute error in measurement units (e.g., mm/s for vibration velocity). RMSE (Root Mean Squared Error): More sensitive to large outliers and anomalous vibration spikes. MAPE (Mean Absolute Percentage Error): Relative error expressed as a percentage, useful for business reports [1, 4, 8].

Stage 5. Cloud Architecture and Containerization (Cloud Deployment) To ensure horizontal scalability and availability, the system is deployed to the cloud: Containerization: The application (FastAPI backend and Streamlit frontend) is packaged into Docker containers. Data Storage: Raw vibration data is stored in cloud storage services (AWS S3, Google Cloud Storage), while prediction results and metrics are stored in time-series databases (InfluxDB, TimescaleDB).

Deployment: Serverless or managed container services are used. They automatically allocate resources: as the load increases, the cloud spins up new container

instances. GPU-based training: Cloud instances with GPU support are utilized for periodic model retraining [1, 4, 8].

Stage 6. Research and practical application in computer engineering. The final stage of the research involves testing the system against real-world benchmarks (such as the well-known NASA IMS Bearing Dataset or data from a specific industrial facility): Comparative analysis: Running LSTM and GRU models on identical vibration data using the same hyperparameters. Performance evaluation: Comparison based on accuracy metrics (MAE/RMSE) and training/inference speed.

Predictive diagnostics: Based on the generated forecast, the system identifies the point at which vibration levels will exceed critical thresholds (in accordance with ISO 10816 standards), enabling the calculation of the machine's Remaining Useful Life (RUL) and the timely issuance of maintenance alerts [1, 4, 8].

The developed software suite operates as a cross-platform desktop application based on a two-tier architecture.

The presentation layer (interface) facilitates interactive user engagement and the visualization of analytical data, while the business logic layer (computational module) handles data import, neural network training, and the generation of forecasts. This clear separation of responsibilities isolates the interface components from the algorithmic logic, significantly simplifying future system modernization and scaling.

3.3 Improving the sequential data processing procedure in the system

The sequential information processing pipeline within the system comprises the following stages: automatic retrieval of time series via the Yahoo Finance API structuring data into a pandas.

To facilitate inter-level interaction, an asynchronous queue mechanism has been integrated: upon completion of the model training procedure, the computational component encapsulates the output data into a `ModelResult` object and delegates it to the `result_queue`, while the main GUI thread retrieves this object via cyclic polling using the `root.after()` method. This architectural approach isolates the time-consuming neural network training processes, preventing the blocking of the Tkinter main event loop. Consequently, the user interface remains continuously responsive and stable while resource-intensive background operations are being executed.

List of the software system's main components and their functional purpose:

- `VibrationForecastApp` (Class, `main.py`) - the application's main class responsible for window initialization, GUI component construction, launching the background training thread, and visualizing vibration diagnostics results and remaining useful life (RUL) forecasts;

- `ModelResult` (Dataclass, `forecasting.py`) - a data structure designed to store the training results of a single model, including actual and predicted vibration

velocity/acceleration values, accuracy metrics, future forecasts, and the loss curve;

- `fetch_vibration_data` (Function, `forecasting.py`) - a function for loading raw time-series data from vibration sensors (via industrial CSV files, time-series databases, or IoT gateways) and performing initial filtering;

- `extract_vibration_features` (Function, `processing.py`) - a module for calculating vibration signal diagnostic features in the time and frequency domains (RMS, crest factor, kurtosis, FFT);

- `train_and_evaluate` (Function, `forecasting.py`) - a function implementing the full workflow for LSTM/GRU models, covering feature normalization (MinMaxScaler), sliding window sequence generation, neural network training, quality metric calculation, and predictive wear trend generation;

- `result_queue` (`queue.Queue`, `main.py`) - an asynchronous queue designed to safely transfer analysis and training results from the background computational thread to the main GUI thread, preventing the interface from freezing.

The system consists of two Python modules: `main.py` and `forecasting.py`. This division adheres to the single responsibility principle: the first module handles only the GUI, while the second manages the computational logic. Separating the interface from the calculation logic simplifies future code maintenance and testing. An analysis of the BTC/USD time series for the last five years revealed no missing values in the target `'Close'` variable. Nevertheless, this validation procedure remains an integral part of the pipeline, as data may arrive in a fragmented state due to network disruptions or technical errors on the market data provider's API side.

4. Discussion of research results

To enable multi-user access to the forecasting system and eliminate the need for local deployment of the Python interpreter on client workstations, a shift to a client-server architecture involving the migration of the computational core to cloud infrastructure has been justified. This architectural paradigm allows the algorithmic logic developed for the `'forecasting.py'` module to remain entirely intact, while transforming the graphical user interface into a web-based frontend or a "thin client" implementation.

The cloud deployment of the analytics platform is carried out through a sequential three-stage process, beginning with the build phase, which involves compiling the Docker software image and subsequently publishing it to a centralized container registry. During the operational phase, the integrated load balancer evenly distributes client requests among active system instances, while a monitoring subsystem based on Prometheus or Cloud Monitoring continuously tracks response latency and error rates.

The creation of decentralized forecasting systems in a multi-cloud environment is complicated by data management issues, security vulnerabilities, and network latency.

Key challenges stem from information reconciliation, platform heterogeneity, and high loads on communication channels.

5. Conclusions

1. A comprehensive analysis entails a detailed examination of recommendations regarding the operation of distributed forecasting models based on cloud platforms. The limited attention paid to this aspect in recent literature has made the development of decentralized forecasting tools a key trend in contemporary research.

2. Based on the research results, a distributed software system for predictive time-series analytics utilizing GRU and LSTM recurrent neural network models and featuring integrated support for cloud deployment was developed and comprehensively analyzed for industrial vibration diagnostics applications. The set objective has been fully achieved: an end-to-end pipeline for processing diagnostic data was successfully implemented spanning from the import of raw high-frequency vibration sensor signals to the generation of a predictive wear trend and the calculation of statistical accuracy metrics.

3. As part of the project, a comparative study of two deep learning architectures was conducted using real-world vibration monitoring data from industrial equipment, and a conceptual framework was established for scaling the local prototype into a containerized cloud service.

4. An analytical review has demonstrated that classical statistical methods (specifically ARIMA and exponential smoothing) are effective only for stationary processes, yet they rapidly lose accuracy when analyzing nonlinear, highly dynamic, and noisy fluctuations in vibration velocity and acceleration. In contrast, modern recurrent neural networks (LSTM, GRU) ensure minimal prediction error for non-

stationary vibration time series associated with the degradation of mechanical components. A critical analysis of existing software confirmed a lack of available tools for the comparative rapid analysis of neural network architectures within a unified computational pipeline. The developed software suite offers practical value and can serve both as a methodological foundation for specialized courses in machine learning and cloud computing and as a functional prototype of an analytical platform for monitoring the actual condition of industrial equipment.

5. Future system modernization will focus on transitioning to a microservices architecture that isolates modules for sensor telemetry collection, background model training, and real-time forecast generation, thereby enabling independent horizontal scaling.

Conflict of interest

The authors declare that they have no conflicts of interest regarding this study whether financial, personal, related to authorship, or of any other nature, that could influence the research and the results presented in this article.

Funding

The study was conducted without financial support.

Data availability

The manuscript has no linked data.

Use of artificial intelligence tools

The authors confirm that they did not use artificial intelligence technologies in the creation of the presented work. The authors confirm that they did not use artificial intelligence technologies in the creation of the presented work.

REFERENCES

1. Hamilton, J. (1994), *Time Series Analysis*, Princeton University Press, Princeton, doi: <https://doi.org/10.2307/j.ctv14jx6sm>
2. Chatfield, C. (2005), "Time-series forecasting", *Significance*, vol. 2, no. 3, pp. 131–133, doi: <https://doi.org/10.1111/j.1740-9713.2005.00117.x>
3. Popovic, D. (2026), *Computational Intelligence in Time Series Forecasting*, Springer-Verlag, London, doi: <https://doi.org/10.1007/1-84628-184-9>
4. Cho, K., Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014), "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation", *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, Doha, Qatar, pp. 1724–1734, doi: <https://doi.org/10.3115/v1/D14-1179>
5. Rashid, T. (2016), *Make Your Own Neural Network* Scotts Valley: CreateSpace, available at: https://app-01-us-dallas-tx.cashcowfarmer.com/virtual-library/YsnnFO/8AD266/make-your-own-neural-network_tariq-rashid.pdf
6. James, G., Witten, D., Hastie, T., and Tibshirani R. (2013), *An Introduction to Statistical Learning: with Applications in R*, Springer, New York, available at: https://www.stat.berkeley.edu/~rabbee/s154/ISLR_First_Printing.pdf
7. Amari, S. (1993), "Backpropagation and stochastic gradient descent method", *Neurocomputing*, vol. 5, no. 4-5, pp. 185–196, doi: [https://doi.org/10.1016/0925-2312\(93\)90006-O](https://doi.org/10.1016/0925-2312(93)90006-O)
8. Moroz, V., and Milovska, K. (2022), "Wavelet analysis and forecasting of financial time series", *Bulletin of the National Technical University «KhPI». Series: Information and Modeling*, vol. 1, no/ 1-2 (7-8), doi: <https://doi.org/10.20998/2411-0558.2022.02.11>
9. Hiransha, M., Gopalakrishnan, E., Menon, V. and Soman, K. (2018), "NSE Stock Market Prediction Using Deep-Learning Models Procedia", *Computer Science*, vol. 132, pp. 1351–1362, doi: <https://doi.org/10.1016/j.procs.2018.05.050>
10. Bengio, Y., Simard, P., and Frasconi, P. (1994), "Learning long-term dependencies with gradient descent is difficult", *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, doi: <https://doi.org/10.1109/72.279181>
11. Hochreiter, S., and Schmidhuber, J. (1997), "Long Short-Term Memory", *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, doi: <https://doi.org/10.1162/neco.1997.9.8.1735>
12. Chung, J., Gulcehre, C., Cho, K. H., and Bengio, Y. (2014), "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling", *arXiv*, arXiv:1412.3555, doi: <https://doi.org/10.48550/arXiv.1412.3555>
13. Chung, J., Gulcehre, C., Cho K. H., and Bengio, Y. (2019), "Comparative Analysis of Forecasting Financial Time Series Using ARIMA, LSTM, and BiLSTM", *arXiv*, arXiv:1911.09512, doi: <https://doi.org/10.48550/arXiv.1911.09512>

14. Franchuk, T., Tyshchenko, D., Desiatko, A., and Karpunin, I. (2025), "Features of accounting digitalization processes", *Galician economic journal*, vol. 95, no 1, pp. 61–66, doi: https://doi.org/10.33108/galicianvisnyk_tntu2025.01.061
15. Kapiton, A., Tyshchenko, D., Franchuk, T., and Desiatko, A. (2025), "Modern Trends in Development and Architectural Features of Computers", *Herald of Khmelnytskyi National University. Technical Sciences*, vol. 359(6.1), pp. 230–234, doi: <https://doi.org/10.31891/2307-5732-2025-359-32>
16. Kapiton, A., Tyshchenko, D., Franchuk, T., Desiatko, A., and Svystun M. (2025), "Advantages and disadvantages of iOS 26", *Information technology and society*, vol. 4 (19), pp. 74–77, doi: <https://doi.org/10.32689/maup.it.2025.4.12>

Received (Надійшла) 15.07.2026

Accepted for publication (Прийнята до друку) 02.09.2026

Publication date (Дата публікації) 25.09.2026

ВІДОМОСТІ ПРО АВТОРІВ / ABOUT THE AUTHORS

Капітон Алла Мирославівна – доктор педагогічних наук, професор, професор кафедри комп'ютерних та інформаційних технологій і систем, Національний університет «Полтавська політехніка імені Юрія Кондратюка», Полтава, Україна;
Alla Kapiton – Doctor of Pedagogical Sciences, Professor, Professor of the Department of Computer and Information Technologies and Systems, Yuriy Kondratyuk Poltava Polytechnic National University, Poltava, Ukraine;
e-mail: kits_seminar@ukr.net; ORCID Author ID: <https://orcid.org/0000-0002-7845-0883>;
Scopus Author ID: <https://www.scopus.com/authid/detail.uri?authorId=60159322400>.

Франчук Тамара Михайлівна – кандидат економічних наук, доцент кафедри інженерії програмного забезпечення та кібербезпеки; Державний торговельно-економічний університет, Київ, Україна;
Tamara Franchuk – Candidate of Economic Sciences, Associate Professor of the Department of Software Engineering and Cybersecurity; State University of Trade and Economics, Kyiv, Ukraine;
e-mail: Tamara.Franchuk@ukr.net; ORCID Author ID: <https://orcid.org/0000-0001-7615-1276>;
Scopus Author ID: <https://www.scopus.com/authid/detail.uri?authorId=57871557200>.

Тищенко Дмитро Олександрович – доцент, кандидат економічних наук, доцент кафедри інженерії програмного забезпечення та кібербезпеки; Державний торговельно-економічний університет, Київ, Україна;
Dmytro Tyshchenko – Associate Professor, Candidate of Economic Sciences, Associate Professor of the Department of Software Engineering and Cybersecurity; State University of Trade and Economics, Kyiv, Ukraine;
e-mail: tyshchenko_d@knute.edu.ua; ORCID Author ID: <https://orcid.org/0000-0002-2193-9012>;
Scopus Author ID: <https://www.scopus.com/authid/detail.uri?authorId=57223189094>.

Захаров Ростислав Геннадійович – доктор філософії, доцент кафедри інженерії програмного забезпечення та кібербезпеки; Державний торговельно-економічний університет, Київ, Україна;
Rostyslav Zakharov – PhD, Associate Professor of the Department of Software Engineering and Cybersecurity; State University of Trade and Economics, Kyiv, Ukraine;
e-mail: r.zakharov@knute.edu.ua; ORCID Author ID: <https://orcid.org/0009-0003-1433-7755>;
Scopus Author ID: <https://www.scopus.com/authid/detail.uri?authorId=58614274000>.

Розробка децентралізованих систем прогнозування на базі мультихмарної інфраструктури

А. М. Капітон, Т. М. Франчук, Д. О. Тищенко, Р. Г. Захаров

Анотація. Актуальність. У дослідженні обґрунтовано потребу в хмарному середовищі, яке забезпечує високий рівень масштабованості системи, уможливаючи інтеграцію нових виробничих потужностей, цехів та обладнання без необхідності капітальних вкладень у власну серверну інфраструктуру. **Предмет дослідження:** процеси розробки та функціонування децентралізованих систем прогнозування на базі мультихмарної інфраструктури. **Мета дослідження:** аналіз сучасних методів розробки децентралізованих систем прогнозування на базі мультихмарної інфраструктури, підвищення їх ефективності, надійності та відмовостійкості. **Завдання дослідження:** дослідити архітектурні аспекти створення децентралізованих систем прогнозування в хмарному середовищі; розглянути ефективність використання розподілених обчислень для технічної діагностики; визначити програмне рішення для прогнозування часових рядів за допомогою рекурентних нейронних мереж; виконати аналіз хмарних платформ і бібліотек машинного навчання для моделювання розподілених часових рядів; виявити ключову проблему для наскрізного аналізу різних нейромережевих архітектур, та сформульовано технічні вимоги для усунення цього обмеження. **Методи дослідження.** Методологія охоплює системний аналіз, математичне моделювання, блокчейн-технології та методи машинного навчання. **Результати дослідження.** Обґрунтовано доцільність застосування хмарних і розподілених архітектур для вирішення діагностичних завдань, що зумовлено специфічними вимогами до обробки великих обсягів даних, які безперервно генеруються первинними датчиками. Визначено, що необхідною умовою є розробка та дослідження програмної системи прогнозування часових рядів на основі обраних моделей нейронних мереж із використанням хмарних обчислень; така система повинна реалізовувати повний цикл обробки даних від їх отримання до формування прогнозу та оцінювання його якості, а отже і для подальшого застосування в галузі комп'ютерної інженерії. Представлено структурований план реалізації наскрізного операційного циклу системи. **Висновки.** Впровадження та розгортання таких систем прогнозування в хмарній інфраструктурі з використанням контейнеризації та оркестрації через керувані сервіси відповідає сучасним інженерним стандартам, що гарантують високу відмовостійкість і горизонтальну масштабованість аналітичних платформ.

Ключові слова: системи прогнозування; хмарна інфраструктура; розподілені системи; синхронізація даних; мережеві затримки; оркестрація даних; машинне навчання; комп'ютерна інженерія; інформаційні технології.