Научное издательство
Профессиональный Вестник

Информационные технологии и безопасность

Выпуск №3/2024

Научный журнал для лучших специалистов отрасли. Внутри: оригинальные работы по ИИ, анализу данных, облачным технологиям и инновациям в сфере ИТ.

ИНДЕКСАЦИИ ЖУРНАЛА

Google Scholar · International Scientific Indexing · Academic Resource Index ResearchBib · Научная электронная библиотека eLIBRARY.RU · CyberLeninka · CiteFactor Academic Scientific Journals

ISSN 3100-444X

support@professionalbulletinpublisher.com

professionalbulletinpublisher.com/

Brasov, Sat Sanpetru, Comuna Sanpetru, Str. Sfintii Constantin si Elena, nr. 6

Scientific publishing house

Professional Bulletin

Information Technology and Security

Issue №3/2024

A scientific journal for the best specialists in the industry. Inside: original works on AI, data analysis, cloud technologies and IT innovations.

INDEXATIONS

Google Scholar · International Scientific Indexing · Academic Resource Index ResearchBib · Научная электронная библиотека eLIBRARY.RU · CyberLeninka · CiteFactor Academic Scientific Journals

ISSN 3100-444X

support@professionalbulletinpublisher.com

professionalbulletinpublisher.com/

Brasov, Sat Sanpetru, Comuna Sanpetru, Str. Sfintii Constantin si Elena, nr. 6

Научное издательство «Профессиональный вестник»

Журнал «Профессиональный вестник. Информационные технологии и безопасность»

Профессиональный вестник. Информационные технологии и безопасность – профессиональное научное издание. Публикация в нем рекомендована практикам и исследователям, которые стремятся найти решения для реальных задач и поделиться своим опытом с профессиональным сообществом. Публикация в журнале подходит для тех специалистов, кто работает и активно развивает передовые ИТ-решения, такие как технологии ИИ, блокчейна, больших данных и другие.

Журнал рецензирует все входящие материалы. Рецензирование – двойное слепое, осуществляется внутренними и внешними рецензентами издательства. Статьи индексируются во множестве международных научных баз, доступ к базе данных журнала открыт для любого читателя. Публикация журнала происходит 4 раза в год.

Сайт издательства: https://www.professionalbulletinpublisher.com/

Выпуск № 3/2024

Жудец Брашов, Румыния

The scientific publishing house «Professional Bulletin»
Journal «Professional Bulletin. Information Technology and Security»

Professional Bulletin. Information Technology and Security is a professional scientific journal. The publication in it is recommended to practitioners and researchers who seek to find solutions to real-world problems and share their experiences with the professional community. The publication in journal is suitable for those specialists who work and actively develop advanced IT solutions, such as AI, blockchain, big data technologies and others.

The journal reviews all incoming materials. The review is double-blind, carried out by internal and external reviewers of the publishing house. Articles are indexed in a variety of international scientific databases, and access to the journal's database is open to any reader. Publication in the journal takes place 4 times a year.

Publisher's website: https://www.professionalbulletinpublisher.com/

Issue №3/2024
Brasov County, Romania

Содержание выпуска

Contents

UDC 004.852

ADAPTIVE MACHINE LEARNING ALGORITHMS FOR STREAM DATA PROCESSING

Baklanov I.
master's degree, Moscow Aviation Institute (Moscow, Russia)

АДАПТИВНЫЕ АЛГОРИТМЫ МАШИННОГО ОБУЧЕНИЯ ДЛЯ ОБРАБОТКИ ПОТОКОВЫХ ДАННЫХ

Бакланов И.Н.
магистр, Московский авиационный институт (Москва, Россия)

Abstract

Adaptive machine learning algorithms are becoming increasingly relevant in the context of processing large volumes of streaming data that are constantly updated and contain significant amounts of noise and outliers. The purpose of this article is to analyze the potential of adaptive algorithms for real-time streaming data processing, with a focus on their application in areas such as financial analytics, the Internet of Things, and cybersecurity. Key methods are discussed, including recurrent neural networks, stochastic gradient descent, and the least squares method, along with their advantages and limitations. Special attention is given to anomaly detection and error prevention using regularization and ensemble methods. The presented results highlight the importance of adaptive algorithms for improving analytical accuracy and system resilience in dynamic environments.

Keywords: adaptive algorithms, machine learning, streaming data, recurrent neural networks, anomalies, cybersecurity.

Аннотация

Адаптивные алгоритмы машинного обучения становятся всё более актуальными в условиях обработки больших объемов потоковых данных, которые постоянно обновляются и содержат значительное количество шума и выбросов. Цель данной статьи – проанализировать возможности адаптивных алгоритмов для обработки потоковых данных в режиме реального времени, с акцентом на их использование в различных отраслях, таких как финансовая аналитика, интернет вещей и кибербезопасность. Рассматриваются основные методы, включая рекуррентные нейронные сети, стохастический градиентный спуск и метод наименьших квадратов, а также их преимущества и ограничения. Отдельное внимание уделено вопросам выявления аномалий и предотвращения ошибок с помощью регуляризации и ансамблевых методов. Представленные результаты подчеркивают важность адаптивных алгоритмов для повышения точности анализа и устойчивости систем в условиях постоянных изменений.

Ключевые слова: адаптивные алгоритмы, машинное обучение, потоковые данные, рекуррентные нейронные сети, аномалии, кибербезопасность.

Introduction

Adaptive machine learning (ML) algorithms are becoming increasingly significant in the context of ever-growing volumes of streaming data. Streaming data is characterized by continuous inflow and high update rates, which make it difficult to process and analyze using traditional methods. As a result, there is a need for algorithms that can adapt to data changes in real-time while maintaining processing accuracy and efficiency. The aim of this article is to analyze the capabilities of adaptive ML algorithms for processing streaming data, focusing on their applications across various industries and potential development prospects.

In recent years, adaptive algorithms have become more popular for analyzing streaming data due to their ability to respond to changes in data structure. This is particularly important when data arrives in real-time and contains a significant amount of noise and outliers. Adaptive methods allow for data processing with minimal resource and time costs, making them ideal for applications such as financial analytics, the Internet of Things (IoT), and cybersecurity. The core principles of adaptive algorithm operation include online learning, parameter adjustment, and fast information processing.

This article will also explore the main methods for implementing adaptive algorithms, such as recurrent neural networks (RNNs) and the least squares method (LSM). Each approach has its advantages and limitations, depending on the type of data and processing objectives.

Main part

Adaptive ML algorithms for streaming data are designed to enable models to update dynamically without complete retraining. One commonly used method is the sliding window approach, where the model analyzes data within a limited time interval. This approach allows the model to adapt to new data, minimizing the risk of becoming outdated and reducing computational resource requirements [1].

RNNs are one of the most widely used methods for handling streaming data, especially when analyzing time sequences. RNNs are effective in systems that require accounting for temporal dependencies, such as IoT. The ability of RNNs to retain information about previous states enables the network to adapt to data changes, making it suitable for real-time data analysis. This capability allows the model to interpret changes occurring in the environment accurately and adjust to current conditions. Stochastic gradient descent (SGD) is often used to optimize models for streaming data. Unlike methods that require data accumulation, SGD updates parameters after each new sample, ensuring high flexibility and model adaptability [2]. This is particularly useful for processing large data streams where maintaining promptness and accuracy is crucial. Table 1 presents a comparison between SGD and mini-batch gradient descent, highlighting the advantages of SGD in real-time applications.

Table 1

Comparative analysis of stochastic and mini-batch gradient descent

Parameter	**SGD**	**Mini-batch gradient descent**
Parameter updates	After each new sample	After accumulating a data batch
Computational load	Low	Medium
Adaptability to streaming data	High	Medium
Memory requirement	Low	Medium
Convergence speed	Fast	Medium
Real-time applicability	High	Limited
Sensitivity to data noise	High	Low
Convergence accuracy	Lower, but fast	More accurate with tuning
Efficiency on large datasets	High	High, but resource-intensive
Computation parallelizability	Low	Medium

Stability under data changes	Low due to sensitivity to each update	More stable due to batch processing

Applying RNNs and SGD for streaming data involves addressing challenges posed by high data volume, such as latency, time-varying characteristics, and significant noise. One approach to enhance RNN resilience to changing conditions is regularization, which reduces model overfitting by adding a complexity penalty. This is especially important for streaming data, where constant weight updates can lead to cumulative errors if the data is unstable [3]. Regularization helps stabilize RNN performance in tasks where accuracy is critical, such as financial forecasting or IoT security monitoring.

To further improve streaming data processing efficiency, adaptive algorithms can use the "gradient clipping" method, which limits the gradient magnitude to prevent it from exceeding a specified threshold. Additionally, ensemble methods like random forests and gradient boosting are widely used to improve model stability and adaptability. Unlike single models, ensembles use a collection of models, each trained on small parts of data or solving specific sub-tasks [4]. Ensembles can enhance prediction accuracy and reduce the likelihood of outliers and anomalies in the data stream.

The following is a simple code example demonstrating the use of SGD for adaptive model training on streaming data:

```
import numpy as np
from sklearn.linear_model import SGDRegressor

# Initializing data and model
data_stream = np.random.rand(1000, 5)  # data stream
targets = np.random.rand(1000)         # target values
model = SGDRegressor(learning_rate='adaptive', max_iter=1, tol=None)

# Step-by-step model adaptation to new data
for i in range(len(data_stream)):
    x = data_stream[i].reshape(1, -1)
    y = np.array([targets[i]])
    model.partial_fit(x, y)

# Displaying model parameters after adaptive training
print("Model parameters:", model.coef_)
```

This example uses the SGDRegressor library from sklearn to implement stochastic gradient descent. The partial_fit parameter allows the model to update as new data arrives without needing full retraining. This approach is useful in scenarios where data arrives in real-time and the model must adapt quickly to changes.

Application of adaptive algorithms for monitoring and anomaly detection

The use of adaptive ML algorithms in monitoring and anomaly detection allows systems to process data in real-time, promptly responding to emerging changes and potential threats. One key area of application for adaptive algorithms is cybersecurity, where timely detection of deviations from normal system behavior is crucial. Such deviations, whether atypical patterns in network traffic or anomalous database queries, may indicate potential attacks or threats [5].

Beyond cybersecurity, adaptive algorithms are widely used for anomaly detection in industrial processes and manufacturing systems. It is important to monitor equipment metric changes to prevent failures and minimize downtime. For instance, ML algorithms can analyze vibration data from machine sensors and detect early signs of wear [6]. Techniques such as SGD and LSM with regularization are often used to increase model robustness against noise and anomalies.

A useful tool in this field is the combination of ML methods with condition monitoring systems, employing algorithms such as principal component analysis (PCA) to reduce data dimensionality and

improve prediction accuracy. This approach allows models to more precisely detect deviations and predict potential failures based on data with numerous variables.

Challenges and future directions in adaptive ML algorithms for streaming data

While adaptive machine learning algorithms provide significant advantages for streaming data processing, several challenges remain that may impact their effectiveness and implementation. One primary challenge is the handling of concept drift – an issue that arises when the statistical properties of the data change over time, which can lead to model degradation. For instance, in financial data or industrial processes, market conditions or equipment performance may shift due to external factors, requiring the algorithm to continually adapt to these changes. Solutions for managing concept drift include implementing periodic retraining or using online learning methods that can dynamically adjust to new patterns without the need for complete model retraining [7].

Another critical challenge is balancing computational efficiency with accuracy. Streaming data typically requires real-time analysis, demanding high-speed data processing with limited computational resources. Adaptive algorithms, while efficient, can still experience delays, especially when handling high-dimensional data [8]. Techniques like dimensionality reduction, including principal component analysis (PCA) and autoencoders, can help streamline data processing by reducing the number of variables the algorithm needs to analyze. However, these techniques may inadvertently omit important features, impacting model accuracy.

Privacy and data security present significant challenges in adaptive ML for streaming data. Many applications, such as those in healthcare or finance, require algorithms that comply with strict privacy regulations, such as GDPR or HIPAA, which restrict data storage and processing. To address this, federated learning has emerged as a promising approach, allowing models to learn from distributed data sources without centralizing data storage [9]. This technique enhances data security but introduces complexities in model coordination and synchronization. Moving forward, improving adaptive algorithms to meet these privacy standards while maintaining processing speed and accuracy will be essential for their broader adoption across various industries.

Conclusion

Adaptive ML algorithms provide unique opportunities for real-time processing of streaming data, ensuring model flexibility and resilience in constantly changing conditions. Through methods such as recurrent neural networks, stochastic gradient descent, and the sliding window approach, algorithms can promptly update model parameters, adapt to new conditions, and maintain analysis accuracy. These approaches help overcome the limitations of traditional methods, which are not always capable of handling large data volumes and rapid variability.

Adaptive algorithms are especially valuable for anomaly detection, which is crucial in areas like cybersecurity and industrial diagnostics. Systems built on adaptive algorithms can automatically detect deviations from normal conditions, alerting to potential threats and ensuring continuous monitoring. Ensemble methods and regularization improve model resilience to noise, enhancing accuracy and reducing false alarms.

Future development of adaptive methods will likely focus on integrating them with advanced technologies such as IoT and big data. These algorithms continue to evolve, increasing their accuracy and predictive capabilities, making them a vital part of modern intelligent data processing systems.

References

1. Akhikyan A.I., Danilyuk S.S. Adaptive random forest machine learning algorithm and its application // Bulletin of Science. 2024. Vol. 1. No.6(75). P. 1393-1402.
2. Voloshin T.A., Zaitsev K.S., Dunaev M.E. Application of adaptive ensemble machine learning methods to time series forecasting tasks // International Journal of Open Information Technologies. 2023. Vol. 11. No.8. P. 57-63.
3. Lebedev I.S. Adaptive application of machine learning models on individual sample segments in regression and classification tasks // Information and Control Systems. 2022. No.3(118). P. 20-30.
4. Nurudinov G.M. Adaptive traffic management in SDN networks using machine learning // Economics and Quality of Communication Systems. 2024. No.1(31). P. 114-122.

5. Kuralenok I., Shchekalev A. GPU in machine learning tasks // Open Systems. Databases. 2013. No.8. P. 44-46.
6. Obukhov A.D., Krasnyansky M.N. Automation of data rerouting process in adaptive information systems using machine learning // Bulletin of Rostov State Transport University. 2020. No.3. P. 106-114.
7. Loeffel P. X. Adaptive machine learning algorithms for data streams subject to concept drifts : diss. Université Pierre et Marie Curie-Paris VI. 2017.
8. Goriparthi R.G. Adaptive Neural Networks for Dynamic Data Stream Analysis in Real-Time Systems // International Journal of Machine Learning Research in Cybersecurity and Artificial Intelligence. 2024. Vol. 15. №1. P. 689-709.
9. Gomes H.M., Read J., Bifet A., Barddal J.P., Gama J. Machine learning for streaming data: state of the art, challenges, and opportunities // ACM SIGKDD Explorations Newsletter. 2019. Vol. 21. No.2. P. 6-22.

UDC 004.6

INTEGRATION OF MACHINE LEARNING IN BIG DATA MANAGEMENT SYSTEMS

Shamsiev A.
Tashkent Institute of Chemical Technology (Tashkent, Uzbekistan)

ИНТЕГРАЦИЯ МАШИННОГО ОБУЧЕНИЯ В СИСТЕМЫ УПРАВЛЕНИЯ БОЛЬШИМИ ДАННЫМИ

Шамсиев А.Д.
Ташкентский химико-технологический институт (Ташкент, Узбекистан)

Abstract

This article examines methods for integrating machine learning (ML) into big data management systems to enhance analytical capabilities and optimize data processing. It analyzes algorithms such as decision trees and deep neural networks, which are applied in tasks like data segmentation, clustering, and time series analysis. Special emphasis is placed on forecasting based on historical data, allowing for improved adaptability and accuracy in analytical systems. Challenges related to computational resources and model robustness against noise and changing data are discussed, with proposed solutions. The results highlight the role of ML in enhancing the efficiency and reliability of modern big data management systems.

Keywords: machine learning, big data, time series, neural networks, clustering.

Аннотация

В статье рассматриваются методы интеграции машинного обучения (МО) в системы управления большими данными для повышения их аналитических возможностей и оптимизации процессов обработки информации. Проанализированы алгоритмы, такие как деревья решений и глубокие нейронные сети, которые находят применение в задачах сегментации данных, кластеризации и анализа временных рядов. Отдельное внимание уделено задачам прогнозирования на основе исторических данных, что позволяет повысить адаптивность и точность аналитических систем. Обсуждаются сложности, связанные с вычислительными ресурсами и устойчивостью моделей к шуму и изменчивым данным, предлагаются возможные пути решения. Полученные результаты подчеркивают роль МО в улучшении эффективности и надежности современных систем управления большими данными.

Ключевые слова: машинное обучение, большие данные, временные ряды, нейронные сети, кластеризация.

Introduction

With the increasing volume of data generated from various sources, including the Internet of Things (IoT), social media, and industrial systems, the need for effective big data management systems is growing. Big data requires high computational power and complex algorithms to ensure their analysis and extraction of valuable insights. In such conditions, the integration of machine learning (ML) methods becomes an integral part of management systems, enabling the automation of data processing and the identification of complex patterns in large data sets. The goal of this article is to explore approaches to integrating ML into big data management systems and evaluate their impact on performance and accuracy.

Machine learning, which is a set of algorithms capable of learning from data and making predictions without explicit programming, opens up new opportunities for big data analysis. The use of ML enhances processing accuracy and speed, as well as uncovers non-obvious dependencies, which is particularly relevant in fields such as medical diagnostics, financial risk prediction, and process optimization. This article will also cover a range of methods, including linear models, decision trees, and neural networks, that have proven effective when working with big data. Additionally, the main challenges such as model optimization under high loads, adaptation to changing data, and ways to increase algorithm robustness to noise will be addressed.

Main part

Integrating ML into big data management systems requires the use of specialized algorithms and architectures that can efficiently process large volumes of information and adapt to changes in data. One such algorithm is decision trees, which classify data based on sequential decisions and are well-suited for processing structured data. Decision trees allow data to be effectively divided into classes, simplifying the process of discovering hidden patterns [1]. However, they can be noise-sensitive for big data, which is why ensemble methods like random forests, which combine multiple trees and enhance error resistance, are often used.

Neural networks, particularly deep neural networks (DNNs), are another crucial component in big data analysis. DNNs with multiple layers of neurons can model complex nonlinear relationships and identify deep connections in data [2]. For instance, when analyzing textual information, DNNs can be used for topic detection and sentiment analysis. Such networks' complexity requires significant computational resources, but using distributed computing and graphics processors (GPUs) helps accelerate the training and data processing process.

Clustering, which is used for data segmentation into groups with similar characteristics, is another important direction in integrating ML into big data management systems. Clustering algorithms like K-means and hierarchical clustering help identify data groups, which is valuable for marketing analysis, bioinformatics, and other areas. Visualizing clustering results aids in better understanding the data structure and drawing conclusions about hidden patterns [3].

Below is an example code demonstrating the application of the K-means method for processing a large data set:

```
import numpy as np
from sklearn.cluster import KMeans

# Generating random data for clustering
data = np.random.rand(1000, 5)

# Setting up and training the K-means model
model = KMeans(n_clusters=5)
model.fit(data)

# Outputting cluster centers
print("Cluster centers:", model.cluster_centers_)
```

In this example, the sklearn library is used to perform clustering with the K-means method. The model divides the data into five clusters, defining each cluster's centers, which allows analyzing the characteristics of each data segment.

ML methods are also widely used for classification and regression tasks in big data management systems. Classification is employed when data needs to be distributed into categories, such as spam detection in emails or object recognition in images [4]. One popular classification method is logistic regression, which, despite its simplicity, is highly effective for binary classification tasks. In big data conditions, this method enables rapid data processing and efficient use of distributed computing.

Another commonly used approach for regression tasks is the support vector machine (SVM) method, suitable for both linearly and non-linearly separable data. With large data volumes, SVM demonstrates high accuracy, although its computational complexity can increase. To manage this

issue, SVM is integrated with dimensionality reduction techniques such as principal component analysis (PCA), which reduces computational load and improves model performance in regression and classification tasks.

Optimizing ML models under big data conditions also requires efficient memory and resource management. One method for optimizing resource use is batch training, where data is divided into small batches, allowing more efficient processing of large data sets [5].

Application of ML in forecasting and time series analysis

Time series analysis is a key task in big data management, especially in areas like finance, medicine, and industry, where forecasting indicator behavior based on historical data is required. Machine learning offers several approaches to time series analysis, enabling models to adapt to changing conditions and provide more accurate forecasts.

Recurrent neural networks (RNNs), which can process sequential data and account for temporal dependencies, are one popular method for time series analysis. RNNs are useful for forecasting tasks such as sales volume prediction, stock market trend analysis, or monitoring equipment state changes [6]. However, traditional RNNs can encounter difficulties when working with long sequences due to the vanishing gradient effect. Enhanced architectures like long short-term memory (LSTM) and attention-based networks are used to solve this problem, allowing better information retention over long intervals.

Another approach to time series analysis is the autoregression method, based on statistical modeling of data dependencies. Autoregressive models are used to predict values based on previous observations and are often combined with ML methods to improve forecast accuracy [7]. For example, an autoregressive model combined with ML algorithms can predict seasonal sales fluctuations or temperature variability.

Below is a code example demonstrating the use of LSTM for time series forecasting:

```
import numpy as np
from keras.models import Sequential
from keras.layers import LSTM, Dense

# Generating time series data
data = np.random.rand(1000, 1)  # sample data

# Preparing data for LSTM
X = []
y = []
time_step = 10
for i in range(len(data)-time_step):
    X.append(data[i:i+time_step])
    y.append(data[i+time_step])
X, y = np.array(X), np.array(y)

# Creating LSTM model
model = Sequential()
model.add(LSTM(50, input_shape=(time_step, 1)))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mean_squared_error')

# Training the model
model.fit(X, y, epochs=10, batch_size=32)

# Forecasting
prediction = model.predict(X[-1].reshape(1, time_step, 1))
print("Forecast:", prediction)
```

This example uses the Keras library to build an LSTM model trained on a time series. This model can analyze sequential data and predict values, making it useful for forecasting based on historical data.

Conclusion

This article has reviewed approaches to integrating ML into big data management systems. It has been shown that using ML algorithms such as decision trees, neural networks, and clustering methods enables efficient processing of large data volumes, revealing hidden patterns, and improving analysis quality. The application of these methods in various industries enhances productivity and reduces labor costs in data processing.

Special attention was given to ML's application for time series forecasting and analysis, which is vital in fields like finance, medicine, and industry. Methods such as recurrent neural networks and autoregression demonstrate high accuracy and adaptability when working with sequential data. Using these methods improves forecast accuracy, which is critical under changing data conditions and a highly dynamic external environment.

The future development of ML integration in big data systems suggests further algorithm improvements and resource optimization. Advances in distributed computing, the use of GPUs, and adaptive learning methods open new horizons for more effective analysis and forecasting, making ML an integral part of modern data management systems.

References

1. Fedutinov K.A. Machine learning in decision support tasks for environmental protection management // Engineering Bulletin of the Don. 2021. No. 9(81). P. 100-113.
2. Ibrahim A., Nikolaev A.S., Bogdanova E.L. Application of machine learning methods in the management system of intellectual property based on blockchain technology // Industry: Economics, Management, Technologies. 2019. No. 2(76). P. 9-14.
3. Kuzmich A.A., Gurin I.A. Integration of document classification models into a conference management system // Thermotechnics and Informatics in Education, Science, and Production (TIM'2024). Yekaterinburg, 2024. P. 174-178.
4. Kondrashov D.E. Integration of machine learning into decision support systems // Potential for Sustainable Innovative Development. 2023. P. 148.
5. Davletov A.R. Main difficulties in integrating machine learning into commercial operation // Innovations and Investments. 2023. No. 10. P. 335-339.
6. Bagutdinov R.A., Stepanov M.F. Methods of integration, data reduction, and normalization of heterogeneous and large-scale data processing // International Journal of Open Information Technologies. 2021. Vol. 9. No. 2. P. 39-44.
7. Antonova I.I., Smirnov V.A., Efimov M.G. Integration of artificial intelligence into ERP systems: advantages, disadvantages, and prospects // Russian Journal of Economics and Law. 2024. Vol. 18. No. 3. P. 619-640.

УДК 004.056.5

ДЕЦЕНТРАЛИЗОВАННЫЕ МЕТОДЫ АУТЕНТИФИКАЦИИ ДЛЯ РАСПРЕДЕЛЕННЫХ СЕТЕЙ

Князева А.С.
бакалавр, Дальневосточный федеральный университет
(Владивосток, Россия)

DECENTRALIZED AUTHENTICATION METHODS FOR DISTRIBUTED NETWORKS

Knyazeva A.
bachelor's degree, Far Eastern Federal University
(Vladivostok, Russia)

Аннотация

Статья анализирует децентрализованные методы аутентификации, применяемые в условиях распределённых сетей для повышения безопасности и отказоустойчивости систем. Рассмотрены криптографические методы, такие как асимметричное шифрование и блокчейн, которые обеспечивают защиту данных и предотвращают несанкционированный доступ. Особое внимание уделено вопросам масштабируемости и производительности децентрализованных решений, а также их потенциалу для использования в сетях интернета вещей и других высоконагруженных системах. Данный анализ выделяет как сильные стороны таких подходов, так и их ограничения, что позволяет оценить их перспективность в условиях меняющихся сетевых требований.

Ключевые слова: децентрализованная аутентификация, распределённые сети, блокчейн, криптография, интернет вещей.

Abstract

The article analyzes decentralized authentication methods used in distributed networks to enhance system security and fault tolerance. Cryptographic approaches, including asymmetric encryption and blockchain, are reviewed as means to protect data and prevent unauthorized access. The focus is placed on the scalability and performance of decentralized solutions, as well as their potential in Internet of Things networks and other high-load systems. This analysis highlights both the strengths and limitations of these approaches, offering insights into their viability within evolving network demands.

Keywords: decentralized authentication, distributed networks, blockchain, cryptography, Internet of Things.

Введение

С ростом использования распределенных сетей и технологий интернета вещей (IoT), аутентификация пользователей и устройств становится критически важной задачей для обеспечения безопасности данных и защиты от несанкционированного доступа. Традиционные централизованные методы аутентификации, такие как серверы авторизации и удостоверяющие центры, сталкиваются с рядом ограничений, включая узкие места производительности, высокую зависимость от единого центра контроля и уязвимость к атакам. В условиях распределенной сети, где количество участников и объём данных увеличиваются, необходимо разрабатывать децентрализованные методы аутентификации, которые позволят повысить безопасность и устойчивость сети. Целью данной статьи является изучение

децентрализованных методов аутентификации, подходящих для современных распределенных сетей, с акцентом на их преимущества и потенциальные ограничения.

Современные децентрализованные методы аутентификации основываются на различных концепциях, таких как блокчейн, криптографические протоколы и методы многофакторной аутентификации. Блокчейн, как основа децентрализованного подхода, позволяет хранить записи аутентификации без единого центра, что обеспечивает неизменность данных и высокую устойчивость к модификациям. Использование криптографии, в свою очередь, обеспечивает надежную проверку подлинности участников сети, уменьшая риск подмены данных и предотвращая несанкционированные попытки доступа. Также децентрализованные системы аутентификации могут использоваться в сочетании с многофакторной аутентификацией, добавляя дополнительные уровни безопасности в системы, требующие высокого уровня защиты.

В данной статье будут рассмотрены основные подходы к децентрализованной аутентификации, включая использование криптографических алгоритмов, распределенных идентификационных систем и блокчейновых решений. Особое внимание будет уделено вопросам повышения безопасности и устойчивости распределенных сетей, а также влиянию децентрализованных методов на производительность сети.

Основная часть

Децентрализованные методы аутентификации в распределенных сетях опираются на использование криптографических протоколов и технологий, исключающих единый центр управления и повышающих устойчивость системы к атакам. Асимметричное шифрование играет ключевую роль в обеспечении аутентичности пользователей: каждый участник сети получает пару ключей – публичный и приватный [1]. Публичный ключ используется для шифрования данных, в то время как приватный служит для их расшифровки. Такой подход исключает необходимость хранения ключей в едином центре, снижая вероятность утечки информации и повышая безопасность сети.

Одним из значимых аспектов является применение хэширования, которое позволяет создать уникальные идентификаторы для каждого участника. В распределенной сети эти идентификаторы записываются в блокчейн, обеспечивая неизменность данных и предотвращая их подделку. Важно отметить, что децентрализованные системы аутентификации, использующие хэш-функции, способны работать без передачи личных данных, что особенно актуально в условиях повышенных требований к защите конфиденциальной информации [2]. Технология Proof-of-Identity, основанная на хэшировании, позволяет проверять подлинность участников, что делает сеть устойчивой к попыткам несанкционированного доступа.

Блокчейн-технология занимает центральное место в децентрализованных методах аутентификации, создавая неизменяемый журнал всех действий, связанных с аутентификацией. Каждый новый запрос аутентификации записывается в отдельный блок, который связывается с предыдущими. Такая структура блоков позволяет обеспечить высокий уровень защиты от подделок и изменений. Данные, записанные в блокчейн, невозможно изменить без одобрения всех участников сети, что делает её устойчивой к злонамеренным действиям.

Кроме того, децентрализованные методы аутентификации применяются в сочетании с многофакторной аутентификацией, которая добавляет дополнительные уровни проверки, усиливая безопасность. Это особенно полезно для систем с высоким уровнем конфиденциальности, таких как финансовые и медицинские сети, где требуется защита от подделок и несанкционированного доступа [3].

Для иллюстрации работы децентрализованных методов аутентификации можно предложить схему, демонстрирующую процесс верификации участников в распределённой сети, использующей блокчейн.

Схема может включать следующие этапы [4]:

1. **Запрос аутентификации**: Участник сети отправляет запрос на аутентификацию, прилагая свои криптографические данные (например, хэш-идентификатор, сгенерированный на основе публичного ключа).
2. **Проверка в узлах сети**: Каждый узел распределённой сети проверяет запрос, сверяя идентификатор с записями в блокчейне. В случае совпадения участник получает подтверждение подлинности, иначе запрос отклоняется.
3. **Запись аутентификации**: Если запрос одобрен большинством узлов, он добавляется в блокчейн как новый блок с отметкой времени и информацией об участнике.
4. **Распределение информации**: Новый блок синхронизируется по всей сети, обеспечивая, что все узлы содержат обновленные данные для будущих запросов аутентификации.

Эта схема позволяет визуализировать весь процесс децентрализованной аутентификации, показывая взаимодействие участников с узлами сети и блокчейном.

Преимущества и ограничения децентрализованных методов аутентификации

Применение децентрализованных методов аутентификации предоставляет значительные преимущества для распределённых сетей, обеспечивая высокий уровень защиты данных и устойчивость к атакам. Одним из главных преимуществ является отсутствие единого центра контроля, что исключает возможность атак на центральный сервер и повышает устойчивость всей системы к отказам [5]. Каждый узел в сети действует независимо, что позволяет системе продолжать функционировать даже при отключении некоторых участников. Децентрализованная аутентификация особенно актуальна для распределённых сетей, где данные поступают из множества источников, и требуется надёжное подтверждение подлинности всех участников.

Однако, несмотря на значительные преимущества, децентрализованные методы имеют и свои ограничения. Одним из основных является высокая нагрузка на вычислительные ресурсы. Обработка и проверка данных, распределённых по блокчейн-узлам, требует больших объёмов памяти и времени, особенно в условиях значительного числа участников. Это ограничивает скорость обработки запросов, что может быть критичным в системах, где важна высокая производительность и оперативность. Кроме того, многие методы, использующие блокчейн, сталкиваются с проблемой масштабируемости, так как каждый новый блок увеличивает общий объем данных, что затрудняет обработку в крупных сетях [6].

Другим важным аспектом является обеспечение конфиденциальности данных. В децентрализованных системах записи хранятся в открытых распределённых реестрах, что может привести к потенциальным рискам утечки информации, если криптографические алгоритмы не будут должным образом защищены. Хотя децентрализованные системы могут быть усилены механизмами шифрования, обеспечение полной конфиденциальности данных остаётся сложной задачей, особенно при использовании публичных блокчейнов.

Заключение

В данной статье были рассмотрены ключевые аспекты децентрализованных методов аутентификации для распределенных сетей, их принципы работы и область применения. Применение асимметричного шифрования, хэширования и блокчейн-технологий позволяет создать устойчивую к отказам систему, защищенную от несанкционированного доступа и с высокой степенью безопасности. Такие методы предоставляют значительные преимущества в обеспечении безопасности данных и устраняют зависимость от единого центра контроля.

Вместе с тем, децентрализованные методы сталкиваются с рядом ограничений, включая высокую нагрузку на вычислительные ресурсы и проблемы масштабируемости, что особенно критично в условиях крупных сетей. Несмотря на это, прогрессивные технологии, такие как блокчейн, позволяют минимизировать часть этих недостатков и открывают новые возможности для повышения производительности. В будущем исследование этих методов будет сосредоточено на оптимизации ресурсов и обеспечении конфиденциальности данных.

Децентрализованные методы аутентификации продолжают оставаться важным направлением развития безопасности в распределенных сетях, таких как IoT и блокчейн. Их

потенциал заключается в создании гибких систем защиты, которые смогут адаптироваться к вызовам современных сетевых технологий, обеспечивая надежную и безопасную аутентификацию.

Список литературы

1. Богораз А.Г. Создание методов защиты децентрализованных распределенных социальных сетей // Системный администратор. 2017. №4. С. 92-95.
2. Алпатов А.Н. Аутентификация с использованием блокчейн в децентрализованных приложениях концепции Web 3.0 // Технологии, модели и алгоритмы модернизации науки в современных геополитических условиях. 2022. С. 27-35.
3. Маслобоев А.В., Путилов В.А. Разработка и реализация механизмов управления информационной безопасностью мобильных агентов в распределенных мультиагентных информационных системах // Вестник Мурманского государственного технического университета. 2010. Т. 13. №4-2. С. 1015-1032.
4. Астахова Т.Н., Колбанев М.О., Шамин А.А. Децентрализованная цифровая платформа сельского хозяйства // Вестник НГИЭИ. 2018. №6(85). С. 5-17.
5. Кругликов С.В., Дмитриев В.А., Степанян А.Б., Максимович Е.П. Информационная безопасность информационных систем с элементами централизации и децентрализации // Вопросы кибербезопасности. 2020. №1(35). С. 2-7.
6. Гончар А.А., Морин Ю.Н., Овсянников А.П. Некоторые вопросы федеративной аутентификации в распределенной сети суперкомпьютерных центров // Труды научно-исследовательского института системных исследований Российской академии наук. 2020. Т. 10. №5-6. С. 13-20.

УДК 004.75

ОЦЕНКА НАДЕЖНОСТИ БЛОКЧЕЙН-СЕТЕЙ ПРИ ВЫСОКОЙ НАГРУЗКЕ

Бейшенов Р.М.
Кыргызско-Российский Славянский университет имени Бориса Ельцина (Бишкек, Кыргызстан)

ASSESSMENT OF BLOCKCHAIN NETWORK RELIABILITY UNDER HIGH LOADS

Beishenov R.
Kyrgyz-Russian Slavic University named after Boris Yeltsin (Bishkek, Kyrgyzstan)

Аннотация

В статье рассматриваются методы повышения надёжности блокчейн-сетей в условиях высокой нагрузки. Приводятся примеры реализации алгоритмов консенсуса, таких как Proof-of-Work, Proof-of-Stake и гибридные модели, в сетях Bitcoin, Ethereum и Solana. Обсуждаются альтернативные решения, включая шардирование и протоколы второго уровня, направленные на улучшение масштабируемости и времени отклика сети. Особое внимание уделяется методам, позволяющим снизить энергозатраты и повысить устойчивость к атакам. Представленный анализ подчёркивает значимость выбора алгоритмов и подходов к масштабируемости для обеспечения стабильности и безопасности блокчейн-сетей под высокой нагрузкой.

Ключевые слова: блокчейн, надёжность, Proof-of-Work, масштабируемость, высокая нагрузка.

Abstract

The article examines methods for improving blockchain network reliability under high load conditions. Examples of consensus algorithms such as Proof-of-Work, Proof-of-Stake, and hybrid models in Bitcoin, Ethereum, and Solana networks are presented. Alternative solutions, including sharding and second-layer protocols, are discussed to enhance network scalability and response times. Special attention is given to methods that reduce energy consumption and increase resistance to attacks. The presented analysis emphasizes the importance of algorithm selection and scalability approaches to ensure the stability and security of blockchain networks under heavy load.

Keywords: blockchain, reliability, Proof-of-Work, scalability, high load.

Введение

С развитием технологий распределённых реестров и ростом популярности блокчейн-сетей обеспечение надёжности данных становится одной из ключевых задач для повышения устойчивости системы. Блокчейн-сети, изначально разработанные для децентрализованного хранения информации, сегодня широко используются для обработки и передачи финансовых транзакций, а также в приложениях, требующих повышенной безопасности. Однако с увеличением нагрузки на блокчейн-сеть возникают новые вызовы, включая снижение производительности, задержки в обработке транзакций и потенциальные риски для сохранности данных. В условиях высокой нагрузки критически важно обеспечить надёжность работы сети для сохранения её производительности и стабильности. Целью данной статьи

является оценка надёжности блокчейн-сетей в условиях высокой нагрузки и анализ методов её повышения.

Современные блокчейн-сети, такие как Bitcoin и Ethereum, демонстрируют определённые ограничения в условиях высокой нагрузки, что может повлиять на скорость подтверждения транзакций и безопасность данных. Одной из ключевых проблем является масштабируемость сети, то есть её способность обрабатывать большое количество транзакций за ограниченное время. Решение данной проблемы требует использования дополнительных механизмов, таких как алгоритмы консенсуса, методы шардирования и оптимизация обработки данных, позволяющие снизить нагрузку на отдельные узлы. В данной статье рассматриваются подходы к улучшению надёжности блокчейн-сетей с акцентом на их способность сохранять производительность в условиях возросшей нагрузки.

Помимо масштабируемости, важным аспектом является устойчивость к атакам и сбоям. Высокая нагрузка может повысить уязвимость сети к различным видам атак, таким как атака 51% или атака отказа в обслуживании (DoS). Использование алгоритмов консенсуса, например Proof-of-Work (PoW) и Proof-of-Stake (PoS), а также их модификаций, помогает блокчейн-сетям поддерживать стабильность и снижать вероятность атак.

Основная часть

Надежность блокчейн-сетей под высокой нагрузкой во многом зависит от выбранных алгоритмов консенсуса, которые обеспечивают согласованность данных и защиту от злоумышленников. **Proof-of-Work (PoW)** является одним из первых алгоритмов консенсуса, успешно применённых в блокчейн-сетях, таких как Bitcoin. Данный алгоритм требует выполнения сложных вычислительных задач для подтверждения транзакций, что делает его устойчивым к большинству атак. Однако высокая вычислительная сложность и энергоёмкость PoW создают значительную нагрузку на сеть, что снижает её пропускную способность и увеличивает задержки при подтверждении транзакций. По мере увеличения количества участников и транзакций PoW может оказаться неэффективным, что требует поиска альтернативных решений [1].

Proof-of-Stake (PoS) представляет собой один из методов, призванных решить проблемы масштабируемости и энергоэффективности PoW. В отличие от PoW, где участники соревнуются за возможность подтвердить транзакцию, PoS использует модель, основанную на ставках участников. Это позволяет значительно снизить энергопотребление и повысить пропускную способность сети. Однако PoS также подвержен риску централизации, поскольку крупные участники, обладающие большими ставками, могут оказывать значительное влияние на процесс подтверждения транзакций. Разработка гибридных алгоритмов, комбинирующих PoW и PoS, может повысить надёжность блокчейн-сетей за счёт сохранения безопасности PoW и улучшенной масштабируемости PoS [2].

Метод шардирования применяется для улучшения производительности блокчейн-сетей путём разделения сети на сегменты, каждый из которых обрабатывает свои транзакции независимо. Шардирование позволяет увеличивать пропускную способность сети и распределять нагрузку между узлами, снижая риск перегрузок и улучшая время отклика. Однако шардирование требует внедрения дополнительных механизмов координации между сегментами, чтобы предотвратить дублирование транзакций и сохранить целостность данных. Например, межшардовые транзакции требуют особой обработки, чтобы обеспечить корректное перемещение данных между различными сегментами сети [3].

Сетевые протоколы второго уровня также играют важную роль в повышении надёжности блокчейн-сетей. Протоколы второго уровня, такие как Lightning Network, позволяют обрабатывать транзакции вне основной цепочки блоков, что снижает нагрузку на сеть и уменьшает задержки при подтверждении транзакций. Эти решения особенно полезны для сетей с высокой нагрузкой, поскольку они обеспечивают более быстрое выполнение транзакций и уменьшают затраты на их обработку [4]. Однако работа таких протоколов требует дополнительных мер безопасности, чтобы предотвратить мошенничество и сохранить данные в целостности при перемещении между основным и вторичным уровнями.

Кроме того, важным аспектом надёжности блокчейн-сетей является их устойчивость к различным атакам, таким как атака 51%, атака двойного расходования и атака отказа в обслуживании (DoS). В условиях высокой нагрузки сети становятся более уязвимыми к атакам, что требует внедрения дополнительных мер безопасности. Для повышения защиты блокчейн-сетей используются механизмы автоматического обнаружения аномалий, а также алгоритмы защиты от DoS, которые блокируют подозрительные транзакции и предотвращают перегрузку узлов. Современные алгоритмы консенсуса также включают механизмы защиты от атак 51%, которые усложняют возможность захвата контроля над сетью [5].

Для иллюстрации использования PoW и PoS в блокчейн-сетях можно представить пример их работы с помощью кода, демонстрирующего базовую структуру подтверждения транзакции:

```
class BlockchainNetwork:
    def __init__(self):
        self.chain = []
        self.pending_transactions = []

    def proof_of_work(self, previous_hash):
        nonce = 0
        while not self.valid_proof(nonce, previous_hash):
            nonce += 1
        return nonce

    def valid_proof(self, nonce, previous_hash):
        # Простая проверка для иллюстрации
        return (str(nonce) + previous_hash).startswith('0000')

    def add_transaction(self, transaction):
        self.pending_transactions.append(transaction)

    def confirm_transactions(self, previous_hash):
        proof = self.proof_of_work(previous_hash)
        block = {'transactions': self.pending_transactions, 'proof': proof}
        self.chain.append(block)
        self.pending_transactions = []
```

Этот код демонстрирует базовую реализацию подтверждения транзакций с использованием PoW. В нём создаются блоки, каждый из которых включает транзакции, проверенные с использованием вычислительной задачи (proof of work).

Примеры реализации и оценки надёжности блокчейн-сетей при высокой нагрузке

Блокчейн-сети, такие как Bitcoin и Ethereum, предоставляют наглядные примеры работы алгоритмов консенсуса в условиях высокой нагрузки и сложных требований к безопасности. Сеть Bitcoin, работающая на алгоритме PoW, обеспечивает высокий уровень безопасности за счёт значительных вычислительных ресурсов, необходимых для подтверждения транзакций. Однако с увеличением количества пользователей и транзакций время подтверждения блоков в сети Bitcoin может возрасти до 10 минут или больше [6]. Для решения проблемы масштабируемости разработан Lightning Network – протокол второго уровня, который позволяет обрабатывать транзакции вне основной цепочки блоков. Это решение уменьшает нагрузку на сеть, ускоряя обработку и снижая стоимость транзакций, что особенно важно в условиях пиковых нагрузок.

Ethereum, одна из наиболее популярных блокчейн-платформ, изначально также использовала PoW для достижения консенсуса. Однако с ростом популярности платформы и увеличением нагрузки разработчики Ethereum столкнулись с проблемой масштабируемости и высокой энергоёмкости. В ответ на эти вызовы была инициирована модернизация сети до

Ethereum 2.0 с переходом на алгоритм PoS, который существенно снижает энергозатраты и повышает пропускную способность [7, 8]. Переход на PoS позволил сети обрабатывать больше транзакций в секунду и снизить время подтверждения блоков. Ethereum 2.0 также внедрил шардирование, что позволяет делить сеть на сегменты для улучшения её производительности и устойчивости в условиях высокой нагрузки.

Другим примером является блокчейн-платформа Solana, которая была разработана с акцентом на высокую пропускную способность и низкую задержку при обработке транзакций. Solana использует уникальный алгоритм консенсуса, основанный на Proof-of-History (PoH) — механизме, который предварительно упорядочивает события в сети, что позволяет улучшить эффективность обработки. Данный алгоритм работает в комбинации с Proof-of-Stake, обеспечивая высокую надёжность и скорость обработки транзакций. Solana способна обрабатывать до 65 000 транзакций в секунду, что делает её одной из самых производительных блокчейн-сетей. Преимущество Solana заключается в её способности справляться с высокой нагрузкой без значительного увеличения затрат на вычислительные ресурсы [9].

Эти примеры показывают, как различные блокчейн-сети применяют уникальные методы и алгоритмы для повышения надёжности и масштабируемости. Несмотря на различия в подходах, все рассмотренные сети стремятся минимизировать время подтверждения транзакций и снизить затраты на их обработку.

Заключение

В данной статье были рассмотрены различные подходы к обеспечению надёжности блокчейн-сетей при высокой нагрузке. Примеры существующих блокчейн-сетей, таких как Bitcoin, Ethereum и Solana, продемонстрировали, что выбор алгоритма консенсуса и метод масштабируемости напрямую влияют на производительность сети и её устойчивость к перегрузкам. Использование PoW обеспечивает высокую защиту от атак, но снижает эффективность при высоких нагрузках из-за большого объёма вычислений. В свою очередь, PoS и гибридные модели позволили блокчейн-сетям повысить масштабируемость и снизить энергопотребление, что особенно важно в условиях интенсивного роста числа транзакций.

Альтернативные подходы, такие как шардирование и сетевые протоколы второго уровня, также сыграли важную роль в оптимизации работы сетей. Шардирование позволило распределить нагрузку между сегментами, улучшив пропускную способность и время отклика сети. Протоколы второго уровня, такие как Lightning Network, предоставили возможность обработки транзакций вне основной цепочки, значительно снижая задержки и повышая удобство использования блокчейна в условиях высоких требований к скорости.

Перспективы развития технологий блокчейн лежат в дальнейшей оптимизации алгоритмов консенсуса и улучшении масштабируемости. Использование гибридных моделей и инновационных протоколов способно повысить надежность блокчейн-сетей, сделав их более устойчивыми к высоким нагрузкам и атакам, что открывает новые возможности для их применения в финансовой и других отраслях.

Список литературы

1. Китанин С.С., Смольянов Б.Д., Чемерис О.С. Перспективы применения технологии блокчейн в энергетическом секторе экономики // Дискуссия. 2024. №5(126). С. 51-57.
2. Истомин Е.П., Кирсанов С.А., Леонтьев Д.В. Некоторые аспекты применения блокчейн-технологий в современной экономике // Информационные технологии и системы: управление, экономика, транспорт, право. 2020. №1. С. 88-102.
3. Афанасьев М.Я., Федосов Ю.В., Крылова А.А., Шорохов С.А. Организация киберфизических производственных систем с использованием технологий блокчейн и смарт-контрактов // Известия высших учебных заведений. Приборостроение. 2019. Т. 62. №3. С. 226-234.
4. Обухов В.А. Цифровая безопасность данных в блокчейн-сетях // PEDAGOG. 2023. Т. 6. №10. С. 304-308.

5. Нурмухаметов Р.К., Степанов П.Д., Новикова Т.Р. Технология блокчейн: сущность, виды, использование в российской практике // Деньги и кредит. 2017. №12. С. 101-103.
6. Руденко Е.А. Понятие системы блокчейн // Проблемы современных интеграционных процессов и пути их решения. 2016. С. 163-164.
7. Соловьев А. Блокчейн: подводные камни // Открытые системы. СУБД. 2016. №4. С. 20-21.
8. Пескова О.Ю., Половко И.Ю., Захарченко А.Д. Применение блокчейн-технологий в системах электронного документооборота: анализ и программная реализация // Инженерный вестник Дона. 2019. №3(54). С. 42.
9. Румянцев И.А. Блокчейн и право // Право в сфере Интернета. 2018. С. 159-178.

УДК 004.65

АНАЛИЗ ПОВЕДЕНЧЕСКИХ ШАБЛОНОВ ПОТРЕБИТЕЛЕЙ НА ОСНОВЕ БОЛЬШИХ ДАННЫХ

Щербакова Е.П.
бакалавр, Сибирский федеральный университет (Красноярск, Россия)

ANALYSIS OF CONSUMER BEHAVIOR PATTERNS USING BIG DATA

Shcherbakova E.
bachelor's degree, Siberian Federal University (Krasnoyarsk, Russia)

Аннотация

Данная статья посвящена анализу поведенческих шаблонов потребителей с использованием технологий обработки больших данных (Big Data) и методов машинного обучения (МО). Описаны алгоритмы кластерного анализа и классификации, включая K-means и случайный лес, для сегментации аудитории и предсказания поведения пользователей. Проведен частотный анализ распределения данных, позволяющий выявить аномалии и выбросы, что важно для построения точных моделей прогнозирования. Показана значимость различных признаков в анализе, таких как частота посещений и время на странице. Полученные результаты подчеркивают важность использования современных аналитических инструментов для повышения конкурентоспособности и адаптации бизнес-стратегий. Представленные методы позволяют глубже понять поведение потребителей и улучшить персонализацию маркетинговых предложений. В статье рассматриваются перспективы дальнейших исследований для повышения эффективности анализа.

Ключевые слова: большие данные, поведенческие шаблоны, машинное обучение, кластерный анализ, частотный анализ.

Abstract

This article focuses on the analysis of consumer behavior patterns using Big Data technologies and machine learning (ML) methods. It describes clustering and classification algorithms, including K-means and Random Forest, for audience segmentation and user behavior prediction. A frequency analysis of data distribution is conducted to identify anomalies and outliers, which is essential for accurate forecasting models. The significance of various features in the analysis, such as visit frequency and time spent on pages, is highlighted. The results emphasize the importance of modern analytical tools to enhance competitiveness and adapt business strategies. The presented methods allow for a deeper understanding of consumer behavior and improved personalization of marketing offers. The article discusses future research directions to further improve analysis efficiency.

Keywords: big data, behavioral patterns, machine learning, clustering analysis, frequency analysis.

Введение

Современное развитие технологий обработки больших данных (Big Data) позволяет анализировать поведенческие шаблоны потребителей с высокой степенью точности и детализации. Эти данные играют важнейшую роль в формировании стратегий маркетинга и персонализированных предложений, что ведет к увеличению конверсии и удовлетворенности потребителей. Основная цель данной статьи – исследовать методы анализа больших данных

для выявления и интерпретации поведенческих паттернов в сфере потребительского поведения, оценив их эффективность и применимость в современных реалиях.

Использование больших данных позволяет не только фиксировать отдельные действия пользователей, но и выявлять скрытые взаимосвязи между их предпочтениями, интересами и привычками. Применение методов машинного обучения (МО) в данной области значительно ускоряет процессы анализа и повышает их точность. Введение алгоритмов кластеризации и классификации дает возможность сегментировать аудиторию и предсказывать дальнейшие действия потребителей, что является ключевым фактором успеха для компаний, стремящихся увеличить свою конкурентоспособность.

Текущие исследования показывают, что внедрение аналитических решений на основе больших данных способно значительно улучшить понимание предпочтений пользователей. В этой статье рассматриваются основные методы анализа данных, их применение для выявления потребительских паттернов и примеры использования на практике.

Основная часть

Анализ поведенческих шаблонов требует использования сложных алгоритмов, включая методы машинного обучения [1]. Одним из таких методов является алгоритм кластерного анализа, позволяющий группировать данные пользователей по определенным признакам. Кластеризация предоставляет возможность сегментировать аудиторию и анализировать особенности каждой группы. Примером такого подхода может служить алгоритм K-means, который находит центры кластеров и распределяет данные на основе минимального расстояния до этих центров [2].

```
from sklearn.cluster import KMeans
import pandas as pd

# Загрузка данных
data = pd.read_csv('consumer_behavior.csv')

# Применение алгоритма K-means для кластеризации
kmeans = KMeans(n_clusters=5, random_state=42)
clusters = kmeans.fit_predict(data)

# Добавление кластеров в таблицу
data['Cluster'] = clusters
```

Данный код позволяет сегментировать данные по пяти кластерам, что помогает выявить группы потребителей с общими характеристиками. После выполнения кластерного анализа результаты необходимо визуализировать для интерпретации. График распределения кластеров представлен на рисунке 1.

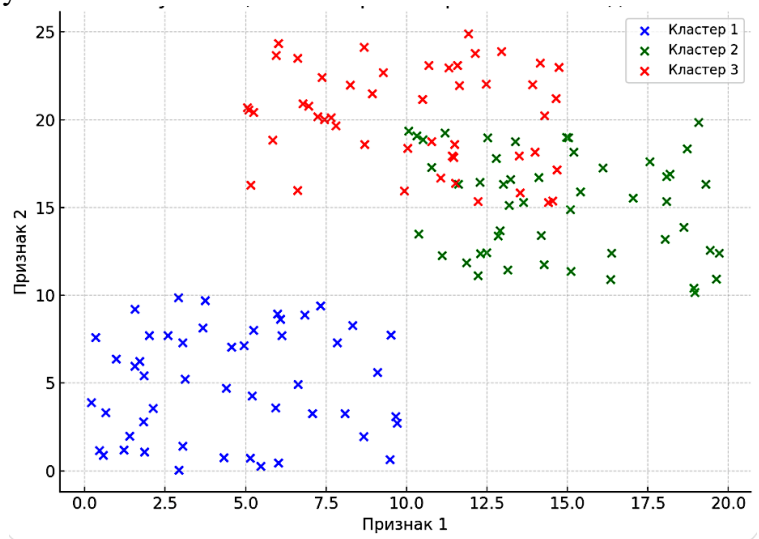

Рисунок 1. Визуализация кластеров на основе поведенческих данных

На графике отображены группы пользователей, выделенные с помощью алгоритма K-means. Каждая точка представляет индивидуального потребителя, а цвет указывает на принадлежность к определенному кластеру.

Кроме кластеризации, важным элементом анализа является применение методов классификации, таких как решающие деревья и случайный лес (Random Forest). Эти алгоритмы позволяют предсказывать поведение потребителей на основе их предыдущих действий [3]. Преимущество метода случайного леса заключается в его способности обрабатывать большие объемы данных с высокой точностью и устойчивостью к переобучению.

Использование модели случайного леса позволяет выявлять ключевые факторы, влияющие на принятие решений потребителями. Например, анализ поведения пользователей в интернет-магазине может показать, что наиболее важными параметрами являются частота посещений сайта и среднее время, проведенное на отдельных страницах [4].

```
from sklearn.ensemble import RandomForestClassifier

# Обучение модели на основе данных
model = RandomForestClassifier(n_estimators=100, random_state=42)
X = data.drop('Purchase', axis=1)  # Удаление целевой переменной
y = data['Purchase']  # Целевая переменная
model.fit(X, y)

# Оценка значимости признаков
importance = model.feature_importances_
```

Значимость признаков помогает лучше понять, какие параметры играют ключевую роль при принятии решения о покупке.

Анализ частотного распределения данных

Анализ частотного распределения данных позволяет оценить плотность и вариативность значений в каждом кластере. Данный подход важен для понимания того, какие диапазоны значений преобладают в данных, и может служить основой для более детального исследования аномалий и отклонений [5]. Распределение данных по Признаку 1 и Признаку 2 (рис. 2) свидетельствует о различиях в плотности групп, что может указывать на особенности поведения каждой группы потребителей.

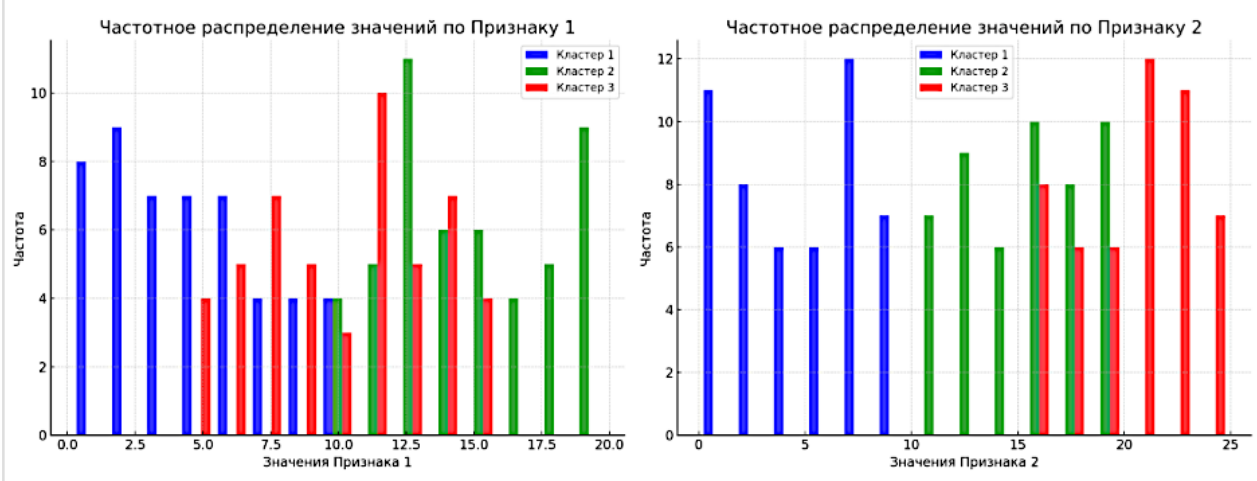

Рисунок 2. Частотное распределение по признакам

Рассмотрение частотных характеристик помогает выявить возможные аномалии, которые могут повлиять на последующий анализ. Например, высокая плотность данных в одном кластере может указывать на сильную корреляцию между признаками, что требует дальнейшего изучения для оценки его влияния на общую стратегию анализа данных.

Частотный анализ данных предоставляет ценную информацию о распределении значений и позволяет глубже понять поведение пользователей. Данные, сгруппированные по

кластерам, могут показывать различия в активности потребителей, их предпочтениях и возможных аномалиях в поведении. Например, если один кластер демонстрирует высокую концентрацию значений в определенном диапазоне Признака 1, это может означать, что пользователи с общими характеристиками предпочитают определенные товары или услуги.

Анализ частотного распределения также позволяет выявить выбросы — значения, сильно отличающиеся от остальных данных [6]. Наличие таких аномальных значений может сигнализировать о потребителях с уникальными поведенческими характеристиками или об ошибках сбора данных. Понимание этих выбросов важно для формирования точных моделей прогнозирования и корректировки стратегии обработки данных.

Кроме того, частотный анализ помогает валидации гипотез о поведенческих паттернах. Например, если предполагалось, что пользователи с определенной характеристикой будут равномерно распределены по кластерам, но анализ показал значительное отклонение, это требует дальнейшего исследования причин такого явления.

Заключение

Анализ поведенческих шаблонов потребителей на основе больших данных играет важную роль в разработке эффективных маркетинговых стратегий и оптимизации бизнес-процессов. Применение методов кластерного анализа и классификации, таких как алгоритм K-means и случайный лес, позволяет сегментировать аудиторию и предсказывать поведение потребителей с высокой точностью. Это способствует лучшему пониманию предпочтений пользователей и более точной персонализации предложений.

Дополнительно, частотный анализ распределения данных помогает выявить аномалии и выбросы, которые могут существенно повлиять на точность моделей прогнозирования. Оценка значимости признаков, таких как частота посещений и время, проведенное на странице, позволяет сосредоточиться на ключевых параметрах, определяющих поведение пользователей.

Внедрение комплексных аналитических подходов, рассматриваемых в данной статье, обеспечивает компаниям конкурентные преимущества, улучшая прогнозирование поведения и адаптацию стратегий под текущие тенденции рынка. Будущие исследования могут быть сосредоточены на совершенствовании методов обработки данных для дальнейшего повышения эффективности анализа и предсказания потребительских паттернов.

Список литературы

1. Коданева С.И. Актуальные проблемы права цифровой экономики // Основные тенденции развития современного права: проблемы теории и практики. 2019. С. 160-163.
2. Коданева С.И. Перспективы развития технологии больших данных: риски нарушения неприкосновенности персональных данных // Социальные и гуманитарные науки. Отечественная и зарубежная литература. Сер. 4, Государство и право: Реферативный журнал. 2019. №3. С. 74-80.
3. Кошмаров М.Ю., Трубецкой А.Ю. Эволюционирование пропаганды. Некоторые аспекты политической практики // Политика и общество. 2018. №8. С. 1-13.
4. Земскова И.А. Дигитализация как фактор повышения качества государственных услуг // Промышленность: экономика, управление, технологии. 2017. №5(69). С. 80-83.
5. Камаев В.А., Финогеев А.Г., Нефедова И.С., Финогеев Е.А. Инструментальные средства "облачного" мониторинга распределенных инженерных сетей // Известия Волгоградского государственного технического университета. 2014. №25. С. 164-176.
6. Родькина А.В. Перспективы анализа больших данных в российском маркетинге // Электронный периодический рецензируемый научный журнал «SCI-ARTICLE. RU». 2018. С. 195.

UDC 004.652

OPTIMIZING STORAGE OF UNSTRUCTURED DATA USING NO-SQL DATABASES

Tulegenova Zh.
bachelor's degree, Kazakh-American University (Almaty, Kazakhstan)

ОПТИМИЗАЦИЯ ХРАНЕНИЯ НЕСТРУКТУРИРОВАННЫХ ДАННЫХ С ПОМОЩЬЮ NO-SQL БАЗ

Тулегенова Ж.С.
бакалавр, Казахско-Американский университет (Алматы, Казахстан)

Abstract

The article discusses the main approaches to storing unstructured data using NoSQL databases, including document-oriented databases (e.g., MongoDB) and key-value stores (e.g., Redis). Examples of practical applications of both technologies are provided, along with code snippets and explanations. The advantages of MongoDB, such as data structure flexibility and horizontal scaling, are highlighted, as well as the benefits of Redis, including high-speed access due to in-memory data storage and support for various data structures. Key differences between these databases and their suitability for different use cases are analyzed. The article emphasizes the importance of choosing the right database according to system requirements. Future research prospects involve integrating NoSQL solutions and creating hybrid architectures to enhance data storage and processing efficiency.

Keywords: NoSQL databases, MongoDB, Redis, data storage, data optimization.

Аннотация

В статье рассматриваются основные подходы к хранению неструктурированных данных с использованием NoSQL баз данных, включая документо-ориентированные базы данных (например, MongoDB) и базы данных типа «ключ-значение» (например, Redis). Описаны примеры применения обеих технологий с приведением кода и объяснением особенностей их использования. Показаны преимущества MongoDB, включая гибкость структуры данных и горизонтальное масштабирование, а также преимущества Redis, такие как высокая скорость доступа благодаря хранению данных в оперативной памяти и поддержка различных структур данных. Обсуждаются ключевые различия между этими базами данных и их применимость для различных сценариев. Статья акцентирует внимание на выборе подходящей базы данных в зависимости от потребностей системы. Перспективы дальнейших исследований связаны с интеграцией NoSQL решений и созданием гибридных архитектур для повышения эффективности хранения и обработки данных.

Ключевые слова: NoSQL базы данных, MongoDB, Redis, хранение данных, оптимизация данных.

Introduction

The modern increase in data volumes generated by various sources, including the Internet of Things (IoT), social networks, and industrial systems, calls for the development of effective solutions for data storage and management. Unlike traditional relational databases, which work well with structured data, processing and storing unstructured data require specialized technologies. One approach that enables efficient management of such data is the use of NoSQL databases, which offer flexibility and scalability when handling various formats and large data volumes. The aim of this

article is to explore optimal approaches for storing unstructured data using NoSQL technologies and to determine how they can contribute to improving the performance and reliability of systems.

NoSQL databases encompass a diverse range of technologies, including document-oriented, graph-based, and wide-column databases. These systems provide flexible data management and adaptation to changing business requirements. One of their key features is horizontal scaling, which makes them particularly useful for processing large volumes of data generated by modern applications and services.

Current research and practical use of NoSQL solutions demonstrate their ability to address challenges in handling unstructured data, such as text files, multimedia elements, and other formats. This article examines the main architectural approaches for optimizing data storage using NoSQL technologies, including their advantages and limitations compared to relational systems.

Main part

One popular approach to storing unstructured data is the use of document-oriented databases such as MongoDB. These databases store data in JSON-like documents, which allows for flexible data management [1]. The example below demonstrates the creation of a collection and the addition of a document to a database using Python.

```
from pymongo import MongoClient

# Connect to the local MongoDB database
client = MongoClient('mongodb://localhost:27017/')

# Create a database
db = client['example_database']

# Create a collection
collection = db['example_collection']

# Add a document to the collection
document = {
   "user_id": 1,
   "name": "Ivan Ivanov",
   "age": 29,
   "preferences": ["movies", "music", "sports"]
}

# Insert the document
collection.insert_one(document)

print("Document successfully added to the collection.")
```

In this code, the pymongo module is used to interact with the MongoDB database. The code includes connecting to the database, creating a new collection, and adding a document with unstructured information. The advantage of this approach is the ability to dynamically change the data structure without compromising system integrity, which is especially useful when dealing with diverse information [2].

The use of document-oriented databases allows for optimized storage by supporting hierarchical and semi-structured data. This approach facilitates the management of data with diverse structures and allows systems to adapt to changing business requirements without needing to modify the entire database [3].

Another example of an effective approach to storing unstructured data is the use of key-value databases like Redis. This database is known for its high performance and ease of use, making it suitable for storing frequently requested data or data requiring quick access. The example below demonstrates how to work with Redis using Python.

```
import redis

# Connect to the local Redis server
client = redis.StrictRedis(host='localhost', port=6379, db=0)

# Add data to Redis
client.set('user:1', '{"name": "Ivan Ivanov", "age": 29, "preferences": ["movies", "music", "sports"]}')

# Retrieve data from Redis
data = client.get('user:1')

# Decode the data
print("Retrieved data:", data.decode('utf-8'))
```

In this code, a connection is established with a local Redis server, and data is stored in JSON string format. The set method is used for data entry, where the key is user:1, and the value is a JSON string. The get method retrieves the data, which is then decoded for display.

Redis is ideal for caching, session management, and temporary data storage that requires quick access. Unlike document-oriented databases, Redis does not offer complex data structures but compensates for this with high operation speed and support for various structures like lists and sets [4].

Advantages and disadvantages of MongoDB

One of the main advantages of MongoDB is the flexibility of its data structure. This database allows for storing data in JSON-like documents, enabling the storage of unstructured information without a strict predefined schema. This approach simplifies data integration and management, especially when the structure requirements frequently change. Another key advantage of MongoDB is its support for horizontal scaling. The system enables sharding, which distributes data across multiple servers, allowing for efficient processing of large data volumes. MongoDB's high read and write performance, aided by indexing and optimization mechanisms, positively impacts data access speed [5].

However, MongoDB has its drawbacks. The absence of a strict data schema can lead to data inconsistencies, particularly in systems with highly complex relationships. This can complicate data management and analysis. Additionally, despite high performance with scaling, MongoDB may require significant resources to maintain performance when handling very large data volumes and complex queries. Effective use of MongoDB capabilities requires specific knowledge and skills for configuration and administration, which may involve additional time and training [6].

Comparison of MongoDB and Redis

MongoDB and Redis are two popular NoSQL databases, each with its strengths and weaknesses, making them suitable for different use cases. MongoDB is a document-oriented database that supports storing data in JSON-like documents. The main advantage of MongoDB is its flexible data structure, which allows unstructured information to be stored and managed without a strict schema [7]. This makes MongoDB an excellent choice for applications where data changes frequently or has a complex hierarchical structure. It also supports horizontal scaling and ensures high performance for read and write operations with the use of indexes.

Redis, on the other hand, is a key-value database known for its extremely high performance due to storing data in memory. This makes Redis ideal for tasks requiring ultra-fast data access, such as caching, session management, message queues, and temporary data. It supports various data structures, including strings, lists, sets, and hashes, which expand its data storage capabilities. However, Redis is less convenient for working with complex hierarchical information compared to MongoDB [8].

The main difference between MongoDB and Redis lies in their approaches to data storage. MongoDB handles larger data volumes better, where data must be persistently stored on disk, while

Redis, due to its in-memory architecture, is faster and more efficient in processing operations but requires additional mechanisms for long-term data storage. Additionally, MongoDB can process more complex queries thanks to its built-in indexing and advanced querying capabilities, whereas Redis is best suited for simple operations with instant response.

Redis has several advantages over MongoDB, particularly when it comes to tasks involving high performance and fast data access. First, the primary advantage of Redis is that it stores data in memory, which ensures extremely fast data access compared to systems using disk storage, including MongoDB [9]. This makes Redis an ideal choice for scenarios where minimal latency is crucial, such as data caching, session management, and message queue processing.

Another significant advantage of Redis is its support for various data types, including strings, lists, sets, and hashes, which allows for efficient handling of different use cases with minimal latency. Due to its architecture, Redis also supports atomic operations, making it suitable for applications where data integrity must be guaranteed during complex operations.

Redis is also known for its simplicity in installation and use. It has an intuitive command-line interface and relatively simple architecture, which facilitates deployment and management. Additionally, Redis can be used as a cache with an automatic data expiration feature (TTL), enabling convenient management of temporary data and ensuring memory is freed as needed [10, 11].

Thus, if the main priority is fast data access and support for simple data structures, Redis offers significant advantages over MongoDB. It is better suited for tasks that require immediate response and where data may not need long-term storage, while MongoDB is more applicable for long-term storage and managing complex data structures.

Conclusion

Optimizing unstructured data storage using NoSQL databases provides new opportunities for enhancing system performance and flexibility. The examples of MongoDB and Redis illustrate how choosing the right database depends on the tasks and data processing speed requirements. MongoDB is suitable for storing data with a changing structure and processing large volumes of information requiring long-term storage. In contrast, Redis, due to its in-memory data storage, ensures instant access, making it ideal for caching and temporary data storage.

The main differences between these systems lie in their data processing approaches: MongoDB offers storage and management of complex hierarchical information, while Redis focuses on high-speed operations and support for simple data structures. This allows system developers and architects to choose the most appropriate solutions based on the specific needs of their applications.

Future research in this field may focus on integrating various NoSQL databases to create hybrid solutions that combine the strengths of different systems. This will enable even greater efficiency and better system adaptation to growing data volumes and processing requirements.

References

1. Ilyin I.V., Ilyashenko V.M. Development of IT architecture in medical organizations based on the introduction of big data technologies // Technological Perspective within the Eurasian Space: New Markets and Points of Economic Growth. 2018. P. 376-382.
2. Kuznetsov S.D., Poskonin A.V. Distributed horizontally scalable solutions for data management // Proceedings of the Institute for System Programming of the RAS. 2013. Vol. 24. P. 327-358.
3. Klemenkov P.A., Kuznetsov S.D. Big data: modern approaches to storage and processing // Proceedings of the Institute for System Programming of the RAS. 2012. Vol. 23. P. 143-158.
4. Burlov V.G., Gryzunov V.V., Sipovich D.E. Adaptive availability management in a geo-information system using fog computing // International Journal of Open Information Technologies. 2021. Vol. 9. No.9. P. 74-87.
5. Seghier N.B., Kazar O. Performance benchmarking and comparison of NoSQL databases: Redis vs MongoDB vs Cassandra using YCSB tool // 2021 International Conference on Recent Advances in Mathematics and Informatics (ICRAMI). IEEE, 2021. P. 1-6.

6. Matallah H., Belalem G., Bouamrane K. Evaluation of NoSQL databases: MongoDB, Cassandra, HBase, Redis, Couchbase, OrientDB // International Journal of Software Science and Computational Intelligence (IJSSCI). 2020. Vol. 12. No.4. P. 71-91.
7. Punia Y., Aggarwal R. Implementing information system using MongoDB and Redis // International Journal of Advanced Trends in Computer Science and Engineering. 2014. Vol. 3. No.2. P. 16-20.
8. Abu Kausar M., Nasar M., Soosaimanickam A. A study of performance and comparison of NoSQL databases: MongoDB, Cassandra, and Redis using YCSB // Indian Journal of Science and Technology. 2022. Vol. 15. No.31. P. 1532-1540.
9. Sánchez R.A.G., Bernal D.J.M., Parada H.D.J. Security assessment of NoSQL MongoDB, Redis, and Cassandra database managers // 2021 Congreso Internacional de Innovación y Tendencias en Ingeniería (CONIITI). IEEE, 2021. P. 1-7.
10. Sen P.S., Mukherjee N. An ontology-based approach to designing a NoSQL database for semi-structured and unstructured health data // Cluster Computing. 2024. Vol. 27. No.1. P. 959-976.
11. Shantharajah S.P., Maruthavani E. A survey on challenges in transforming No-SQL data to SQL data and storing in cloud storage based on user requirement // International Journal of Performability Engineering. 2021. Vol. 17. No.8. P. 703.

УДК 004.312.46

МЕТОДЫ СНИЖЕНИЯ ЭНЕРГОПОТРЕБЛЕНИЯ В УСТРОЙСТВАХ ИНТЕРНЕТА ВЕЩЕЙ

Литвиненко О.А.
специалист, Санкт-Петербургский государственный университет телекоммуникаций имени профессора М.А. Бонч-Бруевича (Санкт-Петербург, Россия)

ENERGY REDUCTION METHODS IN INTERNET OF THINGS DEVICES

Litvinenko O.
specialist degree, Saint Petersburg State University of Telecommunications named after Professor M.A. Bonch-Bruevich (St. Petersburg, Russia)

Аннотация

В данной статье рассматриваются методы и технологии, направленные на снижение энергопотребления в устройствах Интернета вещей (IoT). Описаны примеры энергосберегающих протоколов связи, таких как BLE, Zigbee и LoRaWAN, с характеристиками и основными применениями. Анализируется использование режимов пониженного энергопотребления микроконтроллеров и программных методов оптимизации для сокращения энергозатрат. Особое внимание уделено адаптивным алгоритмам управления частотой процессора и событийно-ориентированной архитектуре, позволяющим устройствам эффективно использовать ресурсы. Представлены рекомендации по выбору методов для различных типов IoT-устройств в зависимости от их задач. Статья подчеркивает значимость комплексного подхода к повышению энергоэффективности и определяет направления дальнейших исследований, направленных на улучшение существующих технологий.

Ключевые слова: IoT, энергосбережение, протоколы связи, программная оптимизация, микроконтроллеры.

Abstract

This article examines methods and technologies aimed at reducing energy consumption in Internet of Things (IoT) devices. Examples of energy-efficient communication protocols such as BLE, Zigbee, and LoRaWAN are described, along with their characteristics and primary applications. The use of low-power modes for microcontrollers and software optimization techniques to reduce energy expenditure is analyzed. Special attention is given to adaptive frequency management algorithms and event-driven architectures, enabling devices to use resources efficiently. Recommendations are provided for selecting methods suitable for various IoT devices based on their specific tasks. The article emphasizes the importance of a comprehensive approach to improving energy efficiency and identifies future research directions for enhancing current technologies.

Keywords: IoT, energy saving, communication protocols, software optimization, microcontrollers.

Введение

В последние годы технологии Интернета вещей (Internet of Things, IoT) развиваются быстрыми темпами, находя применение в различных областях, включая умные дома, промышленную автоматизацию и системы управления городским хозяйством. Основным

вызовом при проектировании IoT-устройств является энергоэффективность, так как многие устройства работают на автономных источниках питания с ограниченной емкостью. Оптимизация потребления энергии в IoT-устройствах критически важна для обеспечения их стабильной работы и продления времени автономного функционирования. Цель данной статьи – рассмотреть методы и технологии, позволяющие снижать энергопотребление в устройствах IoT, и оценить их применимость и эффективность.

Энергоэффективность IoT-устройств напрямую связана с архитектурой и особенностями их компонентов, включая микроконтроллеры, модули беспроводной связи и датчики. Современные подходы к снижению энергопотребления включают оптимизацию аппаратных и программных решений. Разработка энергоэффективных алгоритмов управления и применения технологий энергосбережения играет ключевую роль в повышении общей эффективности устройств. На фоне растущего числа подключенных устройств важность этих методов только возрастает.

В статье анализируются основные стратегии снижения энергопотребления, такие как использование энергосберегающих режимов работы, протоколов связи с низким энергопотреблением и программных методов оптимизации работы процессоров. Особое внимание уделяется сравнительному анализу их эффективности и применению на практике.

Основная часть

Энергосберегающие протоколы связи играют ключевую роль в снижении энергопотребления IoT-устройств, поскольку беспроводная передача данных часто является одним из самых ресурсоемких процессов. Одним из таких протоколов является **Bluetooth Low Energy (BLE),** который был разработан специально для обеспечения минимального энергопотребления при сохранении достаточной скорости передачи данных. BLE использует адаптивные интервалы передачи и функции пониженной активности, позволяя устройствам переходить в спящий режим в периоды бездействия, что значительно снижает энергозатраты. Этот протокол особенно популярен в носимых устройствах и датчиках для умных домов [1].

Еще одним примером является **Zigbee**, который поддерживает создание энергосберегающих сетей с низкой скоростью передачи данных. Zigbee использует механизм передачи на основе пакетов с минимальной задержкой и возможностью гибкого перехода в режимы сна. Данная технология часто применяется в устройствах автоматизации зданий и системах интеллектуального освещения, где важна высокая надежность передачи данных при низком энергопотреблении [2].

Протокол **LoRaWAN** (Long Range Wide Area Network) также заслуживает внимания благодаря своей способности обеспечивать широкую зону покрытия с минимальными затратами энергии. Он позволяет передавать данные на большие расстояния с низкой скоростью, что особенно полезно для удаленных IoT-устройств, таких как сельскохозяйственные датчики и системы мониторинга окружающей среды. LoRaWAN использует технологии модуляции с расширенным спектром, что способствует энергоэффективной передаче данных и продлению времени автономной работы устройства [3].

Другим важным подходом к снижению энергопотребления в IoT-устройствах является использование режимов пониженного энергопотребления микроконтроллеров. Многие современные микроконтроллеры оснащены функциями управления энергопотреблением, которые позволяют переключаться между различными режимами работы в зависимости от нагрузки. Например, микроконтроллеры могут переходить в режимы сна или ожидания, когда устройство не активно, а при необходимости быстрого выполнения операций возвращаться в активный режим. Это позволяет значительно экономить заряд батареи, увеличивая срок службы устройства.

Примером использования этих возможностей служат микроконтроллеры с поддержкой режима Deep Sleep, который отключает большинство периферийных устройств и уменьшает потребление до минимального уровня. Данные методы особенно полезны в устройствах с датчиками, которые собирают информацию с определенной периодичностью. Например,

система мониторинга температуры может оставаться в режиме сна большую часть времени, активируясь только для выполнения измерений и передачи данных.

Программная оптимизация также играет важную роль в повышении энергоэффективности IoT-устройств. Разработка алгоритмов, которые минимизируют использование процессора и уменьшают количество вычислительных операций, способствует снижению энергозатрат. Примером может служить использование адаптивных алгоритмов, которые позволяют устройству изменять частоту обновлений данных в зависимости от изменений в окружающей среде. Это снижает нагрузку на процессор и уменьшает потребление энергии [4].

Примеры программной оптимизации

Программная оптимизация в контексте энергоэффективности IoT-устройств включает различные методы, направленные на снижение нагрузки на процессор и минимизацию энергопотребления. Одним из таких методов является **использование событийно-ориентированной архитектуры**, при которой процессор активируется только при возникновении событий, требующих обработки. Например, вместо периодических проверок состояния датчика используется механизм прерываний, который позволяет устройству оставаться в режиме сна и просыпаться только по мере необходимости.

Еще одним примером программной оптимизации является **адаптивное управление частотой процессора**. Эта техника предполагает динамическое изменение тактовой частоты микроконтроллера в зависимости от текущих вычислительных задач. При низкой нагрузке частота процессора уменьшается, что позволяет значительно сократить энергопотребление. В операционных системах реального времени для IoT часто применяются встроенные функции управления частотой процессора, что позволяет автоматически регулировать мощность устройства [5].

Оптимизация использования памяти и ресурсов также важна для энергосбережения. **Сокращение операций ввода-вывода** помогает минимизировать взаимодействие с периферийными устройствами и тем самым снижает энергозатраты. Например, в приложениях, где требуется передача данных по сети, целесообразно группировать данные и отправлять их пакетами вместо частых одиночных сообщений.

Использование энергоэффективных алгоритмов обработки данных также способствует снижению энергопотребления. Простейший пример – это применение алгоритмов компрессии данных перед их передачей по сети, что позволяет сократить объем передаваемой информации и, как следствие, уменьшить энергозатраты на передачу [6].

Ниже представлена таблица 1 с характеристиками популярных энергосберегающих протоколов для IoT-устройств.

Таблица 1

Характеристики энергосберегающих протоколов связи для IoT-устройств

Протокол	**Максимальная скорость передачи данных**	**Дальность передачи**	**Энергопотребление**	**Основные применения**
BLE (Bluetooth Low Energy)	До 2 Мбит/с	До 100 метров	Низкое	Носимые устройства, умные дома
Zigbee	До 250 Кбит/с	До 100 метров	Низкое	Автоматизация зданий, системы освещения

LoRaWAN	До 50 Кбит/с	До 15-20 км	Очень низкое	Сельскохозяйственные датчики, мониторинг окружающей среды
Wi-Fi HaLow	До 347 Мбит/с	До 1 км	Среднее	Промышленные системы, датчики с большим объемом данных

Таблица, приведенная выше, демонстрирует ключевые характеристики энергосберегающих протоколов связи, применяемых в IoT-устройствах. Сравнение параметров, таких как скорость передачи данных, дальность действия и уровень энергопотребления, позволяет выбрать наиболее подходящий протокол для конкретного применения. Например, BLE и Zigbee обеспечивают низкое энергопотребление и применяются в устройствах с ограниченным радиусом действия, таких как датчики и носимые устройства. В то же время LoRaWAN выделяется своей способностью передавать данные на большие расстояния с минимальными затратами энергии, что делает его идеальным для распределенных систем мониторинга. Wi-Fi HaLow, обладая высокой скоростью передачи, подходит для приложений с увеличенным объемом данных, таких как промышленные датчики, требующие более широкого охвата и производительности [7].

Заключение

Снижение энергопотребления в устройствах IoT является одной из ключевых задач при проектировании современных систем, обеспечивающих автономность и надежность их работы. Рассмотренные методы, включая использование энергосберегающих протоколов связи, оптимизацию программного обеспечения и управление энергопотреблением микроконтроллеров, демонстрируют высокую эффективность в повышении энергоэффективности устройств. Примеры протоколов BLE, Zigbee и LoRaWAN показывают, что выбор подходящего решения зависит от требований к дальности передачи, скорости и уровня энергопотребления.

Оптимизация работы микроконтроллеров и программное управление ресурсами устройства позволяет значительно снизить потребление энергии. Использование событийно-ориентированной архитектуры и адаптивного управления частотой процессора помогает поддерживать баланс между производительностью и энергозатратами. Это особенно важно в системах, где требуется длительное автономное функционирование.

Перспективы дальнейших исследований могут включать разработку гибридных решений, комбинирующих преимущества различных методов и технологий. Это позволит создать более сложные и энергоэффективные системы, которые смогут отвечать растущим требованиям современных приложений Интернета вещей.

Список литературы

1. Мутханна А.С.А. Модель интеграции граничных вычислений в структуру сети «воздух-земля» и метод выгрузки трафика для сетей интернета вещей высокой и сверхвысокой плотности // Труды учебных заведений связи. 2023. Т. 9. №3. С. 42-59.
2. Воробьев А.И., Колбанев А.М., Колбанев М.О. Модель оптимизации энергопотребления умными вещами // Известия СПбГЭТУ «ЛЭТИ». 2015. №7. С. 46.
3. Шиков С.А. Проблемы информационной безопасности: интернет вещей // Инженерные технологии и системы. 2017. Т. 27. №1. С. 27-40.
4. Шешалевич В.В. LPWAN–низкопотребляющие сети большого радиуса действия. Связь для интернета вещей // Безопасность информационных технологий. 2017. Т. 24. №3. С. 7-17.
5. Черняк Л. Интернет вещей: новые вызовы и новые технологии // Открытые системы. СУБД. 2013. №4. С. 14-18.
6. Довгаль В.А., Довгаль Д.В. Анализ проблем обеспечения информационной безопасности беспроводных сенсорных сетей и методов обеспечения безопасности интернета

вещей // Вестник Адыгейского государственного университета. Серия 4: Естественно-математические и технические науки. 2021. №1(276). С. 75-83.
7. Ядровская М.В., Поркшеян М.В., Синельников А.А. Перспективы технологии интернета вещей // Advanced Engineering Research (Rostov-on-Don). 2021. Т. 21. №2. С. 207-217.

УДК 004.415.25

СТРАТЕГИИ МОДУЛЬНОГО ТЕСТИРОВАНИЯ ДЛЯ СОСТАВНЫХ ПРИЛОЖЕНИЙ

Касаткин В.П.
бакалавр, Самарский национальный исследовательский университет имени академика С.П. Королёва (Самара, Россия)

MODULAR TESTING STRATEGIES FOR COMPOSITE APPLICATIONS

Kasatkin V.
bachelor's degree, Samara National Research University named after academician S.P. Korolev (Samara, Russia)

Аннотация

В статье рассматриваются стратегии модульного тестирования, используемые для проверки составных приложений. Описаны инструменты, такие как JUnit, PyTest, NUnit, Jest и Mocha, каждый из которых обладает своими особенностями и преимуществами. Особое внимание уделено инструменту Mocha, который используется для тестирования JavaScript-приложений, включая асинхронные операции. Приведены практические примеры использования Mocha для тестирования модулей и API с поддержкой библиотек, таких как supertest. Рассмотрены подходы к написанию тестов с использованием ассертов и функциональных методов, что помогает автоматизировать процесс тестирования и повысить качество кода. Статья подчеркивает значимость модульного тестирования в современных проектах и необходимость выбора подходящих инструментов для обеспечения стабильности и надежности приложений.

Ключевые слова: модульное тестирование, Mocha, составные приложения, автоматизация тестирования, инструменты тестирования.

Abstract

The article discusses strategies for unit testing used to validate composite applications. It describes tools such as JUnit, PyTest, NUnit, Jest, and Mocha, each with unique features and benefits. Special attention is given to Mocha, a tool for testing JavaScript applications, including asynchronous operations. Practical examples of using Mocha for testing modules and APIs with support for libraries such as supertest are provided. Approaches to writing tests using assertions and functional methods are reviewed, highlighting how to automate the testing process and improve code quality. The article emphasizes the importance of unit testing in modern projects and the need for selecting appropriate tools to ensure application stability and reliability.

Keywords: unit testing, Mocha, composite applications, test automation, testing tools.

Введение

В условиях современного программного обеспечения, где сложные приложения состоят из множества компонентов, модульное тестирование становится критически важным этапом процесса разработки. Модульное тестирование направлено на проверку отдельных частей программного кода, чтобы гарантировать их правильную работу до интеграции в общую систему. Это позволяет минимизировать количество ошибок, которые могут возникнуть при совместной работе компонентов. Цель данной статьи – исследовать стратегии модульного тестирования, применяемые к составным приложениям, и оценить их эффективность и применимость.

Модульное тестирование имеет значительные преимущества, включая раннее обнаружение ошибок, снижение затрат на исправление дефектов и обеспечение стабильности кода. В составных приложениях, где взаимодействие между компонентами может быть сложным, важно не только протестировать отдельные модули, но и убедиться, что они корректно работают при интеграции. Использование стратегий модульного тестирования помогает не только выявить ошибки на ранних этапах разработки, но и улучшить структуру кода, делая его более модульным и поддающимся повторному использованию.

Разработка эффективных стратегий модульного тестирования требует понимания архитектуры приложений и особенностей взаимодействия между компонентами. В данной статье рассматриваются ключевые подходы к модульному тестированию составных приложений, включая использование библиотек и инструментов для автоматизации тестирования, а также принципы разработки тестов с учетом специфики модульных систем.

Основная часть

Модульное тестирование в составных приложениях требует использования специализированных инструментов, обеспечивающих удобство и эффективность тестирования. Наиболее популярными инструментами для этих целей являются **JUnit**, **PyTest**, **NUnit**, **Jest** и **Mocha**, каждый из которых обладает своими особенностями и преимуществами.

JUnit – это один из самых распространенных инструментов для тестирования приложений на языке Java. Он предоставляет простой и удобный интерфейс для написания и выполнения тестов, поддерживает аннотации для обозначения тестовых методов и обеспечивает гибкость при разработке тестовых сценариев. JUnit также интегрируется с различными системами сборки, такими как Maven и Gradle, что делает его незаменимым инструментом в процессе разработки крупных Java-проектов [1]. **PyTest** – это мощный инструмент для тестирования программ на языке Python. Он позволяет легко писать и поддерживать тесты благодаря лаконичному синтаксису и встроенной поддержке различных плагинов. PyTest поддерживает параметризацию тестов, что облегчает создание тестов с разными входными данными, а также имеет развитые механизмы для управления фикстурами и исключениями [2]. **NUnit** используется для тестирования приложений на языке C# и других языках, поддерживаемых платформой .NET. Этот инструмент предоставляет обширный функционал для написания модульных тестов, включая поддержку различных ассертов и атрибутов для обозначения методов тестирования. NUnit легко интегрируется с Visual Studio и другими инструментами разработки для .NET, что делает его популярным выбором среди разработчиков на этой платформе [3]. **Jest** является современным инструментом для тестирования JavaScript и особенно популярен среди разработчиков, работающих с библиотеками и фреймворками, такими как React. Jest обеспечивает встроенную поддержку тестирования компонентов, эмуляцию среды выполнения и генерацию отчетов о покрытии кода. Это упрощает процесс тестирования в современных веб-приложениях и помогает разработчикам быстро находить и исправлять ошибки [4]. **Mocha** – еще один популярный инструмент для тестирования JavaScript, предоставляющий гибкость и расширяемость за счет поддержки множества библиотек ассертов, таких как Chai. Mocha используется для написания как синхронных, так и асинхронных тестов, что делает его универсальным инструментом для различных типов приложений [5].

Эти инструменты помогают автоматизировать процесс модульного тестирования, увеличивая скорость и качество проверки кода.

Подробнее о Mocha

Mocha – это инструмент для модульного тестирования JavaScript, который используется для написания и выполнения синхронных и асинхронных тестов. Для начала работы с Mocha необходимо установить его с помощью npm, используя команду npm install --save-dev mocha. После установки можно запускать тесты с помощью команды npx mocha [6, 7].

Написание тестов в Mocha осуществляется с использованием функций describe и it, которые позволяют группировать тесты и определять отдельные тестовые случаи. Пример простого теста выглядит следующим образом:

```
const assert = require('assert');

describe('Математические операции', function() {
  it('Должно правильно складывать два числа', function() {
    assert.strictEqual(2 + 3, 5);
  });
});
```

В данном примере describe используется для описания группы тестов, а it – для определения конкретного тестового случая. Функция assert.strictEqual проверяет, что результат выражения соответствует ожидаемому значению [8].

Для тестирования асинхронного кода Mocha поддерживает использование параметра done, который передается в функцию обратного вызова и вызывается при завершении теста. Альтернативно можно использовать Promise или async/await для упрощения тестирования:

```
describe('Асинхронные операции', function() {
  it('Должно завершаться через заданное время', function(done) {
    setTimeout(function() {
      assert.strictEqual(true, true);
      done();
    }, 1000);
  });

  it('Должно возвращать результат с использованием async/await', async function() {
    const result = await someAsyncFunction();
    assert.strictEqual(result, 'expectedResult');
  });
});
```

Mocha предоставляет гибкие возможности для тестирования асинхронных операций, что делает его универсальным инструментом для проверки различных типов приложений.

Практическое использование **Mocha** в реальных проектах включает тестирование различных компонентов приложений, включая функциональные модули, API и асинхронные операции. Рассмотрим несколько примеров применения Mocha на практике.

Первый пример – тестирование функции, выполняющей математические операции. Это может быть проверка корректности работы функции сложения:

```
const assert = require('assert');
const mathOperations = require('../src/mathOperations');

describe('Тестирование функций математических операций', function() {
  it('Должно правильно складывать два числа', function() {
    const result = mathOperations.add(2, 3);
    assert.strictEqual(result, 5);
  });

  it('Должно возвращать ошибку при некорректных данных', function() {
    assert.throws(() => mathOperations.add(2, 'a'), /Некорректные данные/);
  });
});
```

В данном примере тестируется модуль mathOperations, где проверяется корректность сложения чисел и выброс ошибок при некорректных входных данных.

Второй пример демонстрирует использование Mocha для тестирования API с помощью библиотеки supertest, которая помогает выполнять HTTP-запросы и проверять ответы:

```
const request = require('supertest');
const app = require('../app'); // приложение Express
```

```
describe('Тестирование REST API', function() {
  it('Должно возвращать список пользователей', async function() {
    const response = await request(app).get('/api/users');
    assert.strictEqual(response.status, 200);
    assert(Array.isArray(response.body));
  });

  it('Должно создавать нового пользователя', async function() {
    const newUser = { name: 'Иван', age: 30 };
    const response = await request(app).post('/api/users').send(newUser);
    assert.strictEqual(response.status, 201);
    assert.strictEqual(response.body.name, 'Иван');
  });
});
```

Здесь Mocha используется в связке с supertest для тестирования маршрутов API, чтобы убедиться, что сервер возвращает ожидаемые данные и корректно обрабатывает запросы.

Третий практический пример касается тестирования асинхронных функций. Mocha позволяет использовать async/await, что делает тесты более читабельными:

```
describe('Асинхронные функции', function() {
  it('Должно возвращать результат через определенное время', async function() {
    const result = await asyncFunctionThatDelays();
    assert.strictEqual(result, 'успех');
  });
});
```

Асинхронные тесты в Mocha полезны для проверки работы с базами данных, взаимодействия с внешними API или выполнения операций, требующих времени. Это делает Mocha универсальным инструментом для создания качественных и надежных тестов для веб-приложений и серверной логики.

Заключение

Модульное тестирование играет важную роль в обеспечении надежности и качества современных приложений, особенно составных систем, где компоненты тесно связаны между собой. Использование специализированных инструментов, таких как JUnit, PyTest, NUnit, Jest и Mocha, позволяет разработчикам эффективно автоматизировать проверку кода и быстро находить ошибки на ранних этапах разработки. Инструмент Mocha демонстрирует гибкость и универсальность при тестировании как синхронных, так и асинхронных операций, что делает его востребованным в веб-разработке и серверных приложениях.

Практические примеры использования Mocha показывают, что инструмент удобен для тестирования функциональных модулей, API и асинхронного кода. Применение ассертов и поддержка интеграции с другими библиотеками, такими как supertest, обеспечивают высокую точность проверки работы компонентов и их взаимодействий. Это помогает не только улучшить качество кода, но и ускорить процесс разработки за счет автоматизации тестов.

В заключение можно отметить, что эффективные стратегии модульного тестирования требуют комплексного подхода, включающего выбор подходящих инструментов и методов. Это позволяет создавать надежные и масштабируемые приложения, минимизируя риск ошибок и повышая общую стабильность системы.

Список литературы

1. Застрожнова А.С., Пеньков Н.А. ИТ-сервис с использованием Red Hat JBoss SwitchYard // Информатика: проблемы, методология, технологии. 2018. С. 71-77.

2. Сергеев А.А. Автоматизация модульного тестирования с Atollic TrueVERIFIER для повышения качества встраиваемых приложений // Компоненты и технологии. 2015. №5. С. 86-91.
3. Puri-Jobi S. Test automation for NFC ICs using Jenkins and NUnit // 2015 IEEE Eighth International Conference on Software Testing, Verification and Validation Workshops (ICSTW). IEEE, 2015. P. 1-4.
4. Park J., An S., Youn D., Kim G., Ryu S. Jest: N+1-version differential testing of both JavaScript engines and specification // 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE). IEEE, 2021. P. 13-24.
5. Takeuchi T., Mouri K., Saito S. Mocha: Automatically Applying Content Security Policy to HTML Hybrid Application on Android Device // 2017 Fifth International Symposium on Computing and Networking (CANDAR). IEEE, 2017. P. 503-509.
6. Holliday M.A., Scott A.S. A software development course based on server-side JavaScript // 2016 IEEE Frontiers in Education Conference (FIE). IEEE, 2016. P. 1-5.
7. Доан Н.В., Сафонов В.О. Средства аспектно-ориентированного программирования для разработки Web-приложений в системе Aspect. NET // Вестник Санкт-Петербургского университета. Прикладная математика. Информатика. Процессы управления. 2011. №1. С. 85-105.
8. Мастеров Д. С. Автоматизированная система функционального тестирования web-приложений : дис. Сибирский федеральный университет. 2016.

UDC 004.056.53

DEVELOPMENT OF DATA PROTECTION METHODS FOR DISTRIBUTED CLOUD SYSTEMS

Ryabova N.
bachelor's degree, Gubkin Russian State University of Oil and Gas (Moscow, Russia)

РАЗРАБОТКА МЕТОДОВ ЗАЩИТЫ ДАННЫХ ДЛЯ ДИСТРИБУТИВНЫХ ОБЛАЧНЫХ СИСТЕМ

Рябова Н.М.
бакалавр, Российский государственный университет нефти и газа имени И.М. Губкина (Москва, Россия)

Abstract

This article discusses data protection methods for distributed cloud systems (DCS), including symmetric, asymmetric, and hybrid encryption, multi-factor authentication, and distributed access control systems. Key characteristics of encryption methods are presented, along with an analysis of their advantages and limitations. Notable data breaches in major companies such as Capital One, Facebook, and Uber are described, emphasizing the importance of robust security measures and regular audits. Modern data protection technologies and their applications in the context of increasing cybersecurity threats are explored. The article highlights the importance of a comprehensive approach and the integration of new methods to enhance the reliability of cloud systems.

Keywords: data protection, distributed cloud systems, encryption, multi-factor authentication, access management.

Аннотация

В данной статье рассматриваются методы защиты данных для дистрибутивных облачных систем (ДОС), включая симметричное, асимметричное и гибридное шифрование, многофакторную аутентификацию и распределенные системы управления доступом. Приведены ключевые характеристики методов шифрования и анализ их преимуществ и ограничений. Описаны случаи утечек данных в крупных компаниях, таких как Capital One, Facebook и Uber, что демонстрирует важность надежных мер безопасности и регулярного аудита. Обсуждаются современные технологии защиты данных и их применение в условиях растущих угроз кибербезопасности. Статья подчеркивает значимость комплексного подхода и интеграции новых методов для повышения надежности облачных систем.

Ключевые слова: защита данных, дистрибутивные облачные системы, шифрование, многофакторная аутентификация, управление доступом.

Introduction

With the development of cloud technologies and distributed computing systems, data protection has become increasingly relevant. Modern distributed cloud systems (DCS) provide users with the ability to process and store data on remote servers, enhancing application flexibility and performance. However, these benefits come with new security challenges. Given the high degree of distribution of such systems, ensuring the confidentiality, integrity, and availability of data is a complex task. The goal of this article is to explore modern approaches to developing data protection methods in DCS and to analyze their effectiveness and applicability in various usage scenarios.

The main security threats in distributed cloud systems include unauthorized access, data leakage, and data integrity violations. Various methods are used to prevent these, including cryptographic algorithms, authentication and authorization mechanisms, as well as segmentation and access control strategies. One of the key aspects of data protection is the use of encryption during both data transmission and storage. It is also essential to consider challenges related to encryption key management and security in multi-tenant environments, where the same resources may be used by different users.

The article reviews the main data protection methods in DCS, including symmetric and asymmetric encryption, multi-factor authentication (MFA) mechanisms, and distributed access management systems. The advantages and limitations of these methods are analyzed considering current performance and security requirements.

Main part

Data encryption methods play a crucial role in protecting information transmitted and stored in distributed cloud systems [1]. Various encryption approaches offer different levels of security and performance, allowing the selection of the most suitable method depending on system requirements. Table 1 below presents the characteristics of the main encryption methods.

Table 1

Characteristics of data encryption methods

Encryption Method	Main Algorithms	Advantages	Disadvantages
Symmetric encryption	AES, DES, 3DES	High speed, efficiency with large data volumes	Issues with secure key management
Asymmetric encryption	RSA, ECC	High security level, convenient key transmission	High computational load
Hybrid encryption	SSL/TLS	Combines advantages of symmetric and asymmetric encryption	Complex implementation, infrastructure needs

The table outlines the main data encryption methods, their advantages, and disadvantages. Symmetric encryption is characterized by high speed, making it suitable for handling large volumes of data, but it requires reliable key management methods. Asymmetric encryption provides a high level of security but imposes significant computational loads. Hybrid encryption, represented by algorithms such as SSL/TLS, combines the advantages of both methods, although its implementation can be complex and require developed infrastructure [2].

In modern DCS, data encryption remains the primary method for ensuring confidentiality. Symmetric encryption, using a single key for encryption and decryption, demonstrates high performance and is often applied to secure data during storage [3]. For example, the AES (Advanced Encryption Standard) algorithm is widely used due to its reliability and speed. However, the main issue remains secure key management, especially in distributed systems where keys may be vulnerable to leaks and compromise. To address these issues, distributed key management methods and secure key exchange protocols are implemented.

Asymmetric encryption, which uses a pair of keys (public and private), ensures a higher level of security during data transmission between different network nodes. Algorithms such as RSA and ECC (elliptic curve cryptography) enable encryption with fewer risks associated with key exchange. However, this type of protection requires significant computational resources, which can slow down data processing in real-time and increase latency in system component interaction [4]. As a result, asymmetric encryption is often used to secure communication channels, after which data transmission continues using symmetric encryption.

Hybrid encryption combines the advantages of symmetric and asymmetric methods. This allows asymmetric encryption to be used for key exchange and symmetric encryption for data encryption. The most well-known example of this approach is the use of SSL/TLS protocols, which

ensure data protection during transmission over the network. SSL/TLS is applied everywhere, from websites to corporate systems, to protect data from interception and tampering.

Besides encryption, an essential element of data protection in DCS is multi-factor authentication (MFA). MFA includes additional verification layers, significantly reducing the risk of unauthorized access. The combination of a password and biometric data or one-time codes provides more reliable user authentication, especially in distributed systems.

Access management systems also play a crucial role in data protection. Modern methods include using roles and access policies that define which users or nodes can access certain data. The role of access management can be enhanced by using distributed access control systems based on blockchain technology, where information about access rights is recorded in a distributed ledger, preventing unauthorized changes [5, 6].

An important addition to data protection strategies is the use of encryption protocols during data transmission. Protocols such as IPSec and VPN provide secure tunnels for data transfer between nodes, eliminating the possibility of interception and packet analysis. These technologies are particularly relevant for the transmission of confidential data between different regions where communications pass through public networks.

Real cases of data breaches and their consequences

Real cases of data breaches in distributed cloud systems demonstrate the importance of reliable information protection. One well-known example is the 2019 data breach at **Capital One**. An attacker gained access to information of more than 100 million customers due to a vulnerability in the Amazon Web Services (AWS) cloud infrastructure settings. The main cause of the breach was improper access management and firewall configuration, allowing the attacker to exploit vulnerabilities to gain access and extract data [7].

Another notable case was the data breach at **Facebook**. In 2019, it was discovered that the data of millions of users was publicly available on cloud servers managed by third parties. The cause of the breach was insufficient access management and the lack of strict control measures when working with partners, leading to inadequate protection of user data [8].

A significant incident also occurred with **Uber** in 2016, when attackers gained access to the data of 57 million users and drivers. The main reason was the use of vulnerable access tokens and weak account protection in the cloud infrastructure. Uber paid substantial fines and committed to strengthening data protection measures, including the implementation of stricter access management mechanisms and audits [9].

These cases show that even leading companies can fall victim to data breaches due to insufficient protection of their distributed cloud systems. The main lessons that can be learned from such incidents include the need for proper security configuration, regular audits, the use of encryption, and access control at all levels of cloud infrastructure [10].

Conclusion

Data protection in distributed cloud systems requires a comprehensive approach that considers a variety of threats and the specifics of distributed architecture. The reviewed methods, including symmetric, asymmetric, and hybrid encryption, as well as multi-factor authentication mechanisms and distributed access management systems, confirm their effectiveness in various application scenarios. These methods help not only protect data during transmission and storage but also minimize the risks of unauthorized access.

Practical examples of data breaches in major companies such as Capital One, Facebook, and Uber emphasize the importance of proper security configuration, regular audits, and effective access management strategies. These incidents serve as a reminder that even industry leaders can face breaches if their infrastructure is insufficiently protected. The lessons learned from these cases contribute to increased awareness of the need for enhanced security measures.

Future developments in data protection methods should focus on the integration of new technologies such as blockchain and artificial intelligence to automate monitoring and threat prevention. These approaches will help create more secure and attack-resistant distributed cloud systems.

References

1. Poludenny N.I., Chernysheva A.V. Analysis of existing methods of software copyright protection using steganography // Software Engineering: Methods and Technologies for Developing Information and Computing Systems (PIIVS-2020). 2020. P. 91-96.
2. Khramtsovskaya N.A. The InterPares Trust international project // Record Management. 2014. No.2. P. 85.
3. Poltavtseva M.A., Kalinin M.O. Analysis of access control systems in heterogeneous big data systems // Proceedings of the Institute for System Programming of the RAS. 2024. Vol. 35. No.4. P. 93-108.
4. Gavrilenko I.R., Slivinsky D.V. New NDC distribution opportunities and IoT as tools for the development of global GDS distribution systems // Economics and Business: Theory and Practice. 2024. No.1-1(107). P. 44-49.
5. Cherepenin V.A., Smik N.O., Vorobyov S.P. Integration of cloud, fog, and edge technologies for the optimization of high-load systems // Software Systems and Computational Methods. 2024. No.1. P. 1-9.
6. Rafique A., Van Landuyt D., Beni E.H., Lagaisse B., Joosen W. CryptDICE: Distributed data protection system for secure cloud data storage and computation // Information Systems. 2021. Vol. 96. P. 101671.
7. Xiaoyu W., Zhengming G. Research and development of data security multidimensional protection system in cloud computing environment // 2020 International Conference on Advance in Ambient Computing and Intelligence (ICAACI). IEEE. 2020. P. 67-70.
8. Gupta I., Singh A.K., Lee C.N., Buyya R. Secure data storage and sharing techniques for data protection in cloud environments: A systematic review, analysis, and future directions // IEEE Access. 2022. Vol. 10. P. 71247-71277.
9. Sun P.J. Privacy protection and data security in cloud computing: a survey, challenges, and solutions // IEEE Access. 2019. Vol. 7. P. 147420-147452.
10. Sun Y., Zhang J., Xiong Y., Zhu G. Data security and privacy in cloud computing // International Journal of Distributed Sensor Networks. 2014. Vol. 10. No.7. P. 190903.