https://doi.org/10.31891/csit-2026-1-8

Leonid CHEPEL
PhD Student, Department Computer Engineering,
Taras Shevchenko National University of Kyiv, Kyiv, Ukraine,
e-mail: leonid.chepel@knu.ua
https://orcid.org/0009-0008-4209-8102
Yuriy BOYKO
PhD, Associate Professor, Department of Computer Engineering,
Taras Shevchenko National University of Kyiv, Kyiv, Ukraine,
e-mail: yuriyboyko@knu.ua
https://orcid.org/0000-0003-1417-7424

Received: 03/12/2026
Accepted:20/02/2026
Published:26/03/2026

UDC 004.72

MULTI-FACTOR HYBRID APPROACH FOR BLOCKCHAIN-ENABLED IOT EDGE COMPUTING

The integration of Internet of Things (IoT) devices with edge computing supports applications in industry, healthcare, and smart cities. Edge computing moves data processing closer to the data source, reducing reliance on central cloud infrastructure. However, managing heterogeneous and geographically distributed systems introduces challenges in coordination, software distribution, and trust establishment. Centralized approaches create single points of failure and limit scalability. Blockchain technology addresses trust and coordination issues. Nonetheless, existing implementations lack on-chain coordination with real-time resource metrics and do not support load-aware task assignments.

This work introduces a hybrid blockchain-IoT edge computing platform. The platform integrates multi-factor node selection based on node reputation and resource availability. The developed approach includes timeout management, queue capacity control, thread-safe concurrent transactions, and event-driven task orchestration with staleness detection. Off-chain computation is separated from on-chain coordination. Node selection uses a scoring function that combines success rate with real-time resource metrics. Coordination logic is implemented through Ethereum-based smart contracts, ensuring transparency and immutability. Validation was conducted in a simulated Docker network environment.

Testing shows that nodes with higher scores are selected in all observed cases. However, when multiple tasks are submitted within the same inter-block period, node scores do not update as expected. Selecting nodes based on the following block rather than the current one improves selection reliability. The platform addresses gaps identified in existing literature and can be used in industrial edge computing scenarios.

Keywords: edge computing, blockchain, internet of things, computer systems, decentralized networks

Introduction

The integration of Internet of Things (IoT) devices with edge computing enables efficient solutions in sectors such as industry, healthcare, smart cities, and infrastructure. Edge computing brings computation and data storage closer to the data sources (e.g., IoT devices, local servers) rather than relying solely on a central cloud. Nonetheless, this approach poses challenges in managing devices, coordinating computations, distributing software, and establishing trust among heterogeneous and geographically dispersed systems. Centralized methods introduce single points of failure, limit scalability, and create dependencies on trusted intermediaries. These issues can be solved with the decentralized principles by utilizing blockchain as main coordinator [1].

Blockchain technology offers a practical approach to trust and coordination issues because it uses immutability, transparency, and decentralized consensus [2, 3]. Immutability means that stored data cannot be changed without detection, while decentralized consensus refers to agreement among distributed nodes instead of a single authority. Recent large-scale systematic reviews indicate growing interest in combining blockchain with Internet of Things (IoT) edge systems to improve security in distributed environments. In this context, blockchain's distributed ledger supports tamper-resistant device registries, verifiable audit logs, and coordination protocols that operate without central control. These attributes are especially useful in multi-party IoT settings where trust among participants cannot be assumed in advance.

Edge computing systems require continuous monitoring of metrics. These metrics include CPU usage, memory

consumption, and task queue status which are necessary due to different edge nodes capabilities [4]. Current blockchain-based research implementations lack on-chain coordination with real-time resource metrics. Existing systems also face challenges with multi-factor selection and load-aware automation. As a result, these systems do not fully support self-balancing and require continuous human intervention to distribute workload efficiently [5, 6].

The aim of this study is to develop a hybrid blockchain-IoT edge computing platform that integrates multi-factor node selection, combining node reputation with resource availability metrics. Additional system requirements include achieving timeout management with queue capacity control, supporting thread-safe concurrent transactions with resource monitoring, and implementing fully automated event-driven task orchestration with staleness detection.

The scientific novelty of the research lies in improving the node selection model, which differs from existing approaches by employing a multi-factor score function that combines success rate with real-time CPU load, memory usage, queue depth, metric freshness, and temporal guarantees. This enables simultaneous optimization for reliability and resource availability, preventing task concentration on overloaded nodes.

The practical significance of this work lies in the creation of a lightweight hybrid platform that can be effectively deployed for industrial edge computing scenarios requiring automatic load balancing. The developed solution addresses gaps identified in systematic literature analysis and provides complete automation from task submission through result verification with resource-aware node selection.

Related Works

Blockchain as a peer-to-peer system supports secure coordination in distributed IoT environments. It provides an encrypted, tamper-resistant record that reduces fraud and unauthorized activities in device networks. The shared ledger records transactions in a consistent way across multiple locations, with time-stamped entries that make data flows and device actions traceable and accountable. In addition, smart contracts enable automated transactions and enforce rules for IoT devices and data exchange, which improve operational efficiency. A consensus protocol is a rule set that enables distributed nodes to agree on the current state of the ledger and governs how new blocks are proposed and confirmed, even when some nodes fail or behave maliciously [2, 3].

Edge computing systems use a three-layer model that includes the Device Layer, Edge Layer, and Cloud Layer. The Device Layer contains sensors and IoT devices. The Edge Layer processes data immediately through local servers. The Cloud Layer manages long-term centralized storage. Decentralization changes this structure by removing central authority. This approach enables peer-to-peer collaboration between nodes. Current research [5, 6] identifies three architectural models that integrate blockchain with edge computing:

1. Blockchain-assisted edge computing systems. Blockchain is used to improve security, trustworthiness, and data integrity in edge environments via immutability and decentralized consensus.
2. Edge-assisted blockchain systems where edge infrastructure is used to improve blockchain performance and scalability.
3. Synergistic integration approaches merge elements of the preceding paradigms for targeted use cases such as distributed energy trading, decentralized finance (DeFi), and federated learning.

Research on blockchain-assisted systems focuses on three main areas. These include off-chain computation offloading, secure node management, and resource optimization [7-13].

For instance, in [7] authors use blockchain for secure data storage and verification. Smart contracts handle identity management, data submission, transaction verification. The architecture enables low-latency processing by computing tasks at the edge while maintaining security through blockchain-based verification and immutable transaction logging. However, the system has several limitations. It lacks real-time resource monitoring and queue capacity management. Node selection does not consider device capabilities. The framework provides no guarantees for task completion deadlines.

Authors from [8] propose a lightweight consensus framework for enhancing blockchain scalability in IoT applications using Delegated Proof of Stake consensus protocol. The model demonstrates impressive performance with low latency through elected delegates. However, delegate selection occurs at consensus layer rather than application layer, and does not consider current resource utilization of selected nodes, potentially assigning tasks to overloaded validators.

The study in [9] develops an AI-driven blockchain-enabled secure edge computing architecture with Layer-2 optimizations. A sophisticated security framework is proposed, showing advantages in threat detection and response. Nevertheless, the implementation lacks resource monitoring integration and queue capacity management, risking task assignment to nodes with exhausted CPU or memory resources.

Research [10] targets blockchain-enabled computation offloading for IoT in mobile edge computing. The authors propose task migration mechanisms between edge servers and cloud infrastructure. However, the absence of real-time load metrics prevents dynamic load balancing, and the model cannot detect stale node information that may lead to suboptimal assignments.

In [11], a decentralized lightweight blockchain-based authentication mechanism is proposed for IoT systems. The application of hash-based verification reduces authentication overhead significantly. However, the model does not address computational task orchestration, resource monitoring, or queue management for edge processing workloads.

Recent work [12] proposes secure firmware update mechanisms using blockchain for integrity verification. Validated on embedded device deployments, it ensures update authenticity through cryptographic hashing. However, it targets software distribution rather than computational task management and lacks the resource-aware selection required for balanced edge computing.

Research [6] synthesizes contemporary advances in blockchain-edge integration, including consensus mechanisms, data integrity, and resource management. The review highlights remaining challenges in multi-factor node selection, real-time resource monitoring, queue capacity enforcement, and production-ready systems combining multiple optimization factors.

Authors in [13] propose a blockchain-based edge computing system that uses node reputation for task allocation. Reputation is calculated from the actual task execution time. Nodes with higher reputation values receive task requests with greater probability. The reputation value is dynamically updated based on task completion efficiency, where servers completing tasks faster than the originally assigned server receive higher reputation increases. However, this approach has important limitations. Overloaded nodes may experience increased latency and higher failure rates. As a result, these nodes may lose reputation despite their capability. The model does not use load-aware metrics for node selection. A multi-factor node selection strategy could improve overall performance. In this strategy, reputation would be one factor among several, such as load level and resource availability.

The paper [14] proposes deadline-aware task scheduling for edge-cloud computing. The method uses a priority-based strategy. Tasks are ordered by deadline proximity. Tasks with tighter deadlines receive higher priority. The method achieves over high deadline satisfaction in simulated environments. However, the approach has limitations for blockchain environments. The system relies on a centralized authority called the Edge-Cloud Management Platform. This contradicts blockchain's trustless model. The priority calculation accounts for node resources. These include CPU, memory, and bandwidth through Comprehensive Execution Capability (CEC). However, priority queue maintenance requires O(n log n) computational overhead. This remains unsuitable for on-chain execution.

Authors in [15] develop an energy-aware task offloading framework using deep reinforcement learning (DRL). The framework addresses deadline constraints. The approach is called DRL-E2D. It optimizes energy consumption and deadline satisfaction. The method uses an actor-critic framework. It combines this with K-nearest neighbor (K-NN) action representation. Results show that the agent converges to maximum reward. This convergence occurs in approximately 150 training episodes. Article [16] uses similar DRL approach for 5G networks. The approach balances device battery life with latency constraints. The solution outperforms baseline methods in dynamic environments. However, the authors identify a key limitation. DRL algorithms create computational overhead on resource-constrained IoT devices.

DRL-based task offloading approaches face several significant limitations for blockchain-based systems. Neural network decisions lack interpretability, as the decision-making process operates as a black box and prevents effective auditing. Environment changes require periodic retraining, making new training cycles necessary when conditions change. Forward pass through neural networks creates computational overhead, which makes on-chain execution unsuitable with gas costs proportional to features multiplied by layers. Different training runs produce different policies, creating non-determinism that violates blockchain's deterministic execution requirements.

Summarizing approaches above, it can be concluded that existing solutions typically do not offer an integrated architecture combining: multi-factor node selection with reputation and resource metrics; real-time CPU, memory, and queue monitoring; staleness detection for outdated node information; queue capacity enforcement preventing overload; and complete event-driven automation with load balancing

Methodology

The proposed approach employs a three-layer hybrid approach separating coordination from computation:

1. IoT Device Layer: task requesters submit computational tasks to the blockchain.
2. Blockchain Layer: Ethereum-based smart contracts manage device registration, task submission, node selection, node metrics tracking, and result verification. All coordination logic executes on-chain ensuring transparency and immutability. Blockchain layer fully handles authentication scenarios and provides additional security layer.
3. Edge Node Layer: distributed worker nodes execute computational tasks off-chain. Each node periodically reports resource metrics (CPU utilization, memory usage, queue depth) to the blockchain, enabling resource-aware selection.

The system operates through event-driven automation where task submissions trigger node selection, assignments trigger execution monitoring, and completions trigger node score updates—all coordinated through smart contracts without centralized control.

The complete task lifecycle that follows proposed approach is presented on figure 1.

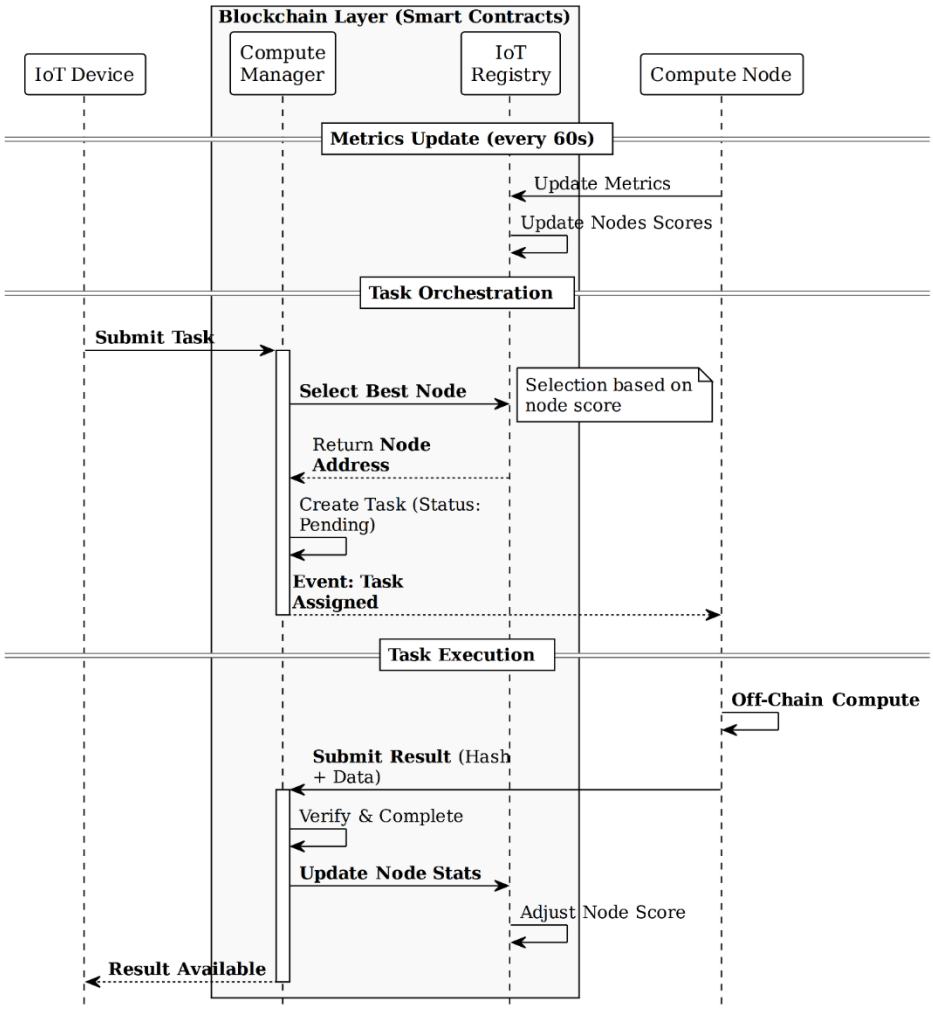

Fig. 1. Blockchain-controlled edge computing task lifecycle

There are 2 main Smart Contracts: *IoTRegistry* is responsible for node registration, node metrics storage, score calculation, and node selection. *ComputeManager* is responsible for task registration, processing, and storing task data — it delegates node selection to *IoTRegistry* when a task is submitted.

Each node reports performance metrics every minute. These reports demonstrate the node remains active. The system uses these metrics to calculate node scores. Task distribution depends heavily on these scores. Nodes with higher scores receive more tasks. The system penalizes nodes that fail to report metrics. When a node misses a metric submission, its score decreases significantly. This mechanism ensures continuous node participation and reliable network operation.

The classical reputation-based score for node selection can be represented as:

$$n_s = \arg\max_{n \in N} \mathrm{R}(n), \tag{1}$$

where n_s – a selected node for pending task; $\mathrm{R}(n)$ – reputation of node n across N available nodes.

The proposed solution is to extend existing approach with multi-factor approach that combines independent dimensions by using multiplicative composition:

$$n_s = \arg\max_{n \in N} \left[\mathrm{R}(n) \cdot \mathrm{F}_{\mathrm{cpu}}(n) \cdot \mathrm{F}_{\mathrm{mem}}(n) \cdot \mathrm{F}_{\mathrm{queue}}(n) \cdot \mathrm{F}_{\mathrm{fresh}}(n) \cdot \mathrm{F}_{\mathrm{time}}(n)\right], \tag{2}$$

where $\mathrm{F}_{\mathrm{cpu}}(n)$ – current CPU availability factor of node n; $\mathrm{F}_{\mathrm{mem}}(n)$– memory availability factor; $\mathrm{F}_{\mathrm{queue}}(n)$– task queue capacity factor; $\mathrm{F}_{\mathrm{fresh}}(n)$ – metrics freshness factor; $\mathrm{F}_{\mathrm{time}}(n)$ – temporal guarantee factor, meaning tasks are finalized in time; the value range of all F functions is [Fmin,1] and $\mathrm{F}_{\mathrm{min}}$ is minimal positive non-zero value.

Multiplicative composition limits compensation - a critical property for resource-constrained systems. In multi-criteria decision making (MCDM) articles approaches are categorized as compensatory, non-compensatory, and partial compensatory based on whether poor performance in one criterion can be offset by high performance in another. The multiplicative approach implemented here aligns with the Weighted Product Model (WPM) [17]. The use of $\mathrm{F}_{\mathrm{min}}$ as a floor value represents a partial compensatory approach as even if all nodes are overloaded, there still should be the best node n_s. The design ensures that resource exhaustion in any dimension causes significant score reduction.

The selection of WPM is based on Table 1 and the constraints of on-chain smart contracts. Running high-complexity computations within a smart contract increases gas usage and reduces performance. In some cases, such computations may not be feasible due to on-chain limitations.

Table 1

Comparison of multi-factor approach methods

Method	Complexity for n nodes and k criteria	Description
Weighted Product Model (WPM)	O(n·k)	Calculates a multiplicative product of criteria scores for each alternative. Poor performance in any single criterion highly reduces the overall score [17].
Weighted Sum Model (WSM)	O(n·k)	Calculates a weighted sum of criteria scores for each alternative. Poor performance in one criterion can be fully offset by strong performance in others [17].
Analytic Hierarchy Process (AHP)	$O(k \cdot n^2)$	Derives priority weights via pairwise comparison matrices. Poor performance in one criterion is fully offset depending on expert-assigned weights [18]. The mathematical complexity of these operations results in high gas usage during on-chain smart contract execution.
Technique for Order Preference by Similarity to Ideal Solution (TOPSIS)	O(n·k)	Ranks alternatives by Euclidean distance to ideal and anti-ideal solutions using vector normalization. Poor performance in one criterion can be compensated by proximity to the ideal in others [19]. The mathematical complexity of these operations results in high gas usage during on-chain smart contract execution.
Game-theoretic auction models	Each round: O(n·k)	Agents bid for resources based on individual utility functions. Incorporating multiple criteria requires complex multi-attribute mechanism design. Multi-round bidding introduces high network latency. It also increases gas costs, as each action requires a separate state transition, requiring off-chain execution [20].
Deep Reinforcement Learning (DRL)	Depends on architecture	A neural network agent learns offloading policies through reward-driven training episodes. Not compatible with on-chain computations [15] [16].

The node reputation score is used as initial basis for overall score calculation and can be presented as:

$$R(n) = \frac{T_s(n)}{T_a(n)} \cdot R_{max}, \quad (3)$$

where $T_s(n)$ – successfully completed tasks count for node n; $T_a(n)$ – all tasks assigned to node n; R_{max} – constant, indicating maximum reputation score.

Other dimensions heavily rely on node system resources and behavior factors including metrics freshness and task execution time:

$$\begin{gathered} F_{cpu}(n) = \max\left(F_{min}, 1 - L_{cpu}(n, \Delta t)\right), \\ F_{mem}(n) = \max\left(F_{min}, 1 - L_{mem}(n, \Delta t)\right), \\ F_{queue}(n) = \max\left(F_{qmin}, 1 - L_{queue}(n, \Delta t)\right), \\ F_{fresh}(n) = \begin{cases} 1 & \text{if metrics} < t_m \text{ old} \\ F_{min} & \text{if metrics} > t_m \text{ old'} \end{cases} \\ F_{time}(n) = \max\left(F_{min}, \min\left(1, \frac{T_{timeout}}{T_{expected}(n)}\right)\right), \\ T_{expected}(n) = T_{avg}(n, \Delta t) \cdot \left(1 + L_{queue}(n, \Delta t)\right) \end{gathered} \quad (4)$$

where $L_{cpu}(n,\Delta t)$, $L_{mem}(n,\Delta t)$, $L_{queue}(n,\Delta t)$ - the time-weighted average of CPU, memory and tasks queue capacity load of node n over a duration Δt; the value range of L-functions is [0,1]; F_{qmin} – minimal acceptable value of $F_{queue}(n)$ when tasks queue is full; F_{min} – minimal positive non-zero value; t_m – metrics refresh time in minutes; $T_{timeout}$ – maximum allowed task execution time, provided by task requester; T_{avg} – average task execution time of node n over a duration Δt.

A proposed multi-factor approach assigns tasks to the most capable nodes with linear complexity. However, when the node reaches its concurrent task limit and F_{qmin} is not small enough to lower score among other nodes, the system should select the next node with the next-highest score that can process the task immediately.

Implementation

The practical part includes the development of smart contracts and Python scripts. The Python scripts represent an edge node and IoT device. System is based on Go-Ethereum project as it is production-tested and includes lightweight clients, designed for computationally limited devices [3]. For tiny devices like IoT sensors a specialized implementation of the web3 library can be used to connect to blockchain nodes [21]. For IoT environment it is recommended to use Proof of Authority consensus protocol [22, 23]. Go Ethereum Clique was used in Docker network with a partial mesh network topology to simulate real-world conditions.

Smart contracts data structures were defined as follows:

```
interface IIoTRegistry {
enum Role { NONE, ADMIN, DEVICE, COMPUTE_NODE }
struct Device {
  bool     active;
  string   softwareVersion;
  string   downloadUrl;
  bytes32  fileHash;
  uint256  registeredAt;
 }

struct ComputeNode {
 bool     active;
 string   endpoint;
 uint256  computeScore;
 uint256  lastUpdate;
 uint256  tasksProcessed;
 uint256  tasksFailed;
 uint256  pendingTasks;
 uint256  avgCpuLoad;
 uint256  avgMemoryLoad;
 uint256  avgQueueLoad;
 uint256  avgTaskTime;
 uint256  taskTimeout;
 uint256  lastMetricsWindow;
 uint256  maxQueueCapacity;
    }
}
interface IComputeManager {
    enum TaskStatus { Pending, Processing, Completed, Failed }
    struct Task {
        address    creator;
        bytes32    taskType;
        address    assignedNode;
        bytes      resultData;
        bytes32    resultHash;
        TaskStatus status;
        uint256    createdAt;
        uint256    completedAt;
    }
}
```

The administrator registers new devices and nodes in the system. IoT devices receive a DEVICE role, while nodes receive a COMPUTE_NODE role. This approach separates responsibilities between different blockchain accounts. IoT devices and nodes maintain separate blockchain accounts. All operations require a signature from the respective account key. Therefore, devices and nodes operate independently using their own credentials.

Each smart contract generates specific events. When a device registers, the IoTRegistry smart contract produces a DeviceRegistered event. When a software update is required, the contract produces a DeviceSoftwareUpdated event. Devices update their software by downloading from the URL provided in the event. The URL requires authorization through a request header. This authorization uses a signature created with the device's blockchain credentials.

Events are captured by nodes or devices through a polling mechanism. A full list of events is presented in Table 2.

Table 2

Blockchain events emitted by the IoTRegistry and ComputeManager smart contracts

Event Name	Description
IoTRegistry	
NodeRegistered	Emitted when a compute node self-registers for the first time.
NodeMetricsUpdated	Emitted after node metrics is updated.
DeviceRegistered	Emitted when an admin registers a new IoT device and assigns it the DEVICE role.
DeviceSoftwareUpdated	Emitted when admin pushes a firmware update record for a device.
ConfigUpdated	Emitted for each scoring parameter changed by an admin
ComputeManager	
TaskSubmitted	Emitted when a device or admin successfully submits a task; includes the node selected by invoking getBestNode() from IoTRegistry.
TaskStatusChanged	Emitted on every status transition: Pending → Processing → Completed/Failed.
ResultSubmitted	Emitted after a node submits a verified result (keccak256 check passed on-chain).
ScriptRegistered	Emitted when an admin registers or updates a computation script.

The node selection algorithm shown in Figure 2 is implemented in the *getBestNode()* method of the *IoTRegistry* smart contract. The algorithm analyzes the metrics of all available nodes and calculates a score for each one. Based on Formula 2, the node with the highest score is selected for the current task.

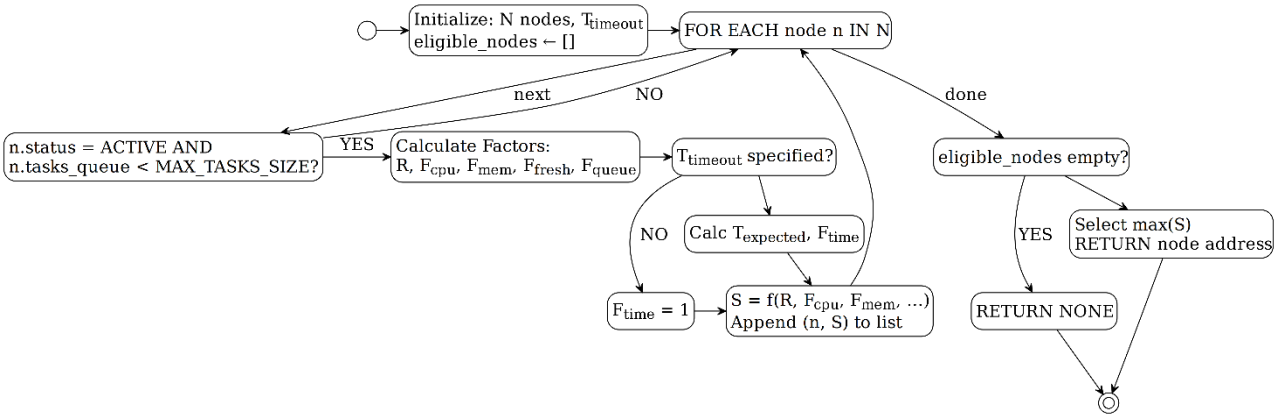

Fig. 2. Node selection algorithm

The Python node uses asynchronous event monitoring to poll for new events every five seconds. This non-blocking approach ensures the system remains responsive during polling intervals. Task management is enforced through a limit of ten parallel tasks running concurrently, preventing resource exhaustion under high load conditions. Resource monitoring is performed in real time using the Python built-in psutil library, which tracks CPU and memory usage. Blockchain nonce management relies on thread-safe mechanisms to handle concurrent transactions without conflict. When a transaction failure occurs, the system automatically resets the nonce. This error recovery process reduces manual intervention and improves overall reliability.

The taskType field of Task entity encodes computation script name that should be executed by the node. The computation script must expose a single synchronous entry point:

```
function compute(input_data: bytes) → dict
    Input: raw task payload from blockchain (JSON-encoded)
    Output: JSON-serialisable dict; on error {error: message}

    params ← decode(input_data)
    result ← execute(params)
    return result
```

The computation script must satisfy several behavioral constraints. It must complete within the number of seconds specified by the task_timeout parameter. Additionally, the function must return a JSON-serializable dictionary in all cases, including error states.

Data integrity verification is based on *keccak256* hash function. Both input and output data are verified by hash.

Results

The test system consisted of a Docker network with ten Go-Ethereum Clique nodes and ten edge nodes. Block time was set to three seconds, and the node polling interval was set to five seconds. Each node script was configured to simulate real-world scenarios with different CPU and memory loads, so nodes reported different metrics. Each task was configured to simulate consumption by 20% of CPU and 10% of memory in simulated metrics, while the actual task was a script with a five-minute delay. Consequently, the reported metrics reflected this additional load, resulting in a lower final score for each node.

A dedicated script was implemented to generate tasks for the test. It created 20 tasks in mini-batches of five tasks each every minute. The script recorded task identifiers and monitored task start and completion events. After all tasks finished, it retrieved metric history, scores, and node information from the blockchain for further analysis. Table 3 shows a system snapshot at a specific point during task assignment.

Table 3

Data captured prior to next task assignment

Node	Reputation	CPU, %	Memory, %	Tasks Assigned	Score (0-1000)
1	1000	1	28	0	720
2	1000	25	40	1	450
3	1000	50	20	0	400
4	1000	34	90	0	66
5	1000	40	20	2	384
6	1000	55	45	0	248
7	1000	29	40	1	383
8	1000	30	43	1	359
9	1000	60	74	0	104
10	1000	12	95	0	44

Testing revealed that when multiple tasks are submitted simultaneously within the same inter-block period, the node score does not update as expected, which can cause node overload. Although score updates are triggered by task submission and also occur automatically every minute, the actual score can change at most once per block. To address this, delayed node assignments should be implemented: the system should select the best node based on the next block rather than the current one. This adds a one-block delay before a task starts, but improves the reliability of node selection.

The score was captured before each task submission. Figure 3 shows the distribution of selected node scores.

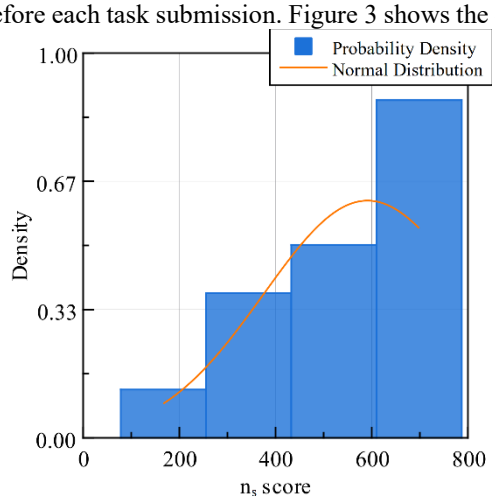

Fig. 3. Selected node score distribution

The results indicate that nodes with higher scores are selected in most cases, which confirms the theoretical approach. The next step is to conduct testing on real devices to cover cases that may not have been observed during simulation testing.

Conclusions

This article addresses its stated objectives by developing and validating a hybrid blockchain-IoT edge computing platform. The proposed platform successfully integrates a multi-factor node selection strategy in a simulated Docker network using Go-Ethereum Clique. The implementation confirms that nodes with higher calculated scores are selected for task allocation in all cases.

The main difference with existing blockchain-based edge implementations lies in the improved node selection model. This process uses a multi-factor scoring function that mathematically combines node reputation with real-time availability metrics, including CPU load, memory usage, queue depth, metrics freshness, and temporal guarantees. The other difference is the hybrid approach that separates off-chain computation from on-chain coordination, utilizing Ethereum-based smart contracts to manage all coordination logic and event-driven automation in a secure and coordinated way.

Further research can optimize the platform's implementation. Specifically, the node selection reliability and score update mechanisms can be improved. For instance, adopting a delayed node assignment strategy—selecting the best node based on the next block rather than the current one—could prevent node overload during massive inter-block task submissions. However, it adds a one-block delay before a task starts.

The next step could be to conduct testing on real devices to cover cases that may not have been observed during simulation testing. Evaluating this hybrid platform in real-world industrial edge-computing scenarios that require automatic load balancing will provide crucial data to validate the solution's effectiveness.

ADDITIONAL INFORMATION

CONTRIBUTIONS OF AUTHORS

Conceptualization, methodology – Leonid Chepel, Yuriy Boyko; formulation of tasks, analysis – Leonid Chepel, Yuriy Boyko; development of the hybrid approach, testing – Leonid Chepel; analysis of results – Leonid Chepel; writing – original draft preparation, writing – review and editing – Leonid Chepel, Yuriy Boyko.

USE OF ARTIFICIAL INTELLIGENCE

In addition to the standard Scopus search, Perplexity AI's research mode was used to search for sources published in the last 5 years. The authors verified the results provided by reviewing the original sources and their content and confirm that it did not influence research conclusions. ChatGPT was used for grammar check and rephrasing. After using this tool/service, the authors reviewed and edited the content and take full responsibility for the content of this publication.

REFERENCES

1. Hossain M., Kayas G., Hasan R., Skjellum A., Noor S., Islam S. M. R., Hossain M., Kayas G., Hasan R., Skjellum A., Noor S., Islam S. M. R. A Holistic Analysis of Internet of Things (IoT) Security: Principles, Practices, and New Perspectives. Future Internet. 2024. Vol. 16, no. 2.

2. Nakamoto S. Bitcoin: A Peer-to-Peer Electronic Cash System | Satoshi Nakamoto Institute [Electronic resource]. 2008. URL: https://nakamotoinstitute.org/library/bitcoin/ (accessed 18.01.2026).

3. Buterin V. Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform [Electronic resource]. 2014. URL: https://ethereum.org/en/whitepaper (accessed 01.11.2025).

4. Giannakopoulos P., Knippenberg B. van, Joshi K. C., Calabretta N., Exarchakos G. Key metrics for monitoring performance variability in edge computing applications. EURASIP Journal on Wireless Communications and Networking. 2025. Vol. 2025, no. 1. P. 38.

5. Xue H., Chen D., Zhang N., Dai H.-N., Yu K. Integration of blockchain and edge computing in internet of things: A survey. Future Generation Computer Systems. 2023. Vol. 144. P. 307–326.

6. Nezami Z., Li Z., Qin C., Banaie F., Khalid R., Pournaras E. Blockchain and Edge Computing Nexus: A Large-scale Systematic Literature Review. 2025.

7. Asaithambi S., Nallusamy S., Yang J., Prajapat S., Kumar G., Rathore P. S. A secure and trustworthy blockchain-assisted edge computing architecture for industrial internet of things. Scientific Reports. 2025. Vol. 15, no. 1. P. 15410.

8. Haque E. U., Abbasi W., Almogren A., Choi J., Altameem A., Rehman A. U., Hamam H. Performance enhancement in blockchain based IoT data sharing using lightweight consensus algorithm. Scientific Reports. 2024. Vol. 14, no. 1. P. 26561.

9. Qu Y., Pokhrel S. R., Garg S., Gao L., Xiang Y. A Blockchained Federated Learning Framework for Cognitive Computing in Industry 4.0 Networks. IEEE Transactions on Industrial Informatics. 2021. Vol. 17, no. 4. P. 2964–2973.

10. Xu X., Zhang X., Gao H., Xue Y., Qi L., Dou W. BeCome: Blockchain-Enabled Computation Offloading for IoT in Mobile Edge Computing. IEEE Transactions on Industrial Informatics. 2020. Vol. 16, no. 6. P. 4187–4195.

11. Khalid U., Asim M., Baker T., Hung P. C. K., Tariq M. A., Rafferty L. A decentralized lightweight blockchain-based authentication mechanism for IoT systems. Cluster Computing. 2020. Vol. 23, no. 3. P. 2067–2087.

12. Witanto E. N., Oktian Y. E., Lee S.-G., Lee J.-H., Witanto E. N., Oktian Y. E., Lee S.-G., Lee J.-H. A Blockchain-Based OCF Firmware Update for IoT Devices. Applied Sciences. 2020. Vol. 10, no. 19.

13. Gao Q., Xiao J., Cao Y., Deng S., Ouyang C., Feng Z. Blockchain-based collaborative edge computing: efficiency, incentive and trust. Journal of Cloud Computing. 2023. Vol. 12, no. 1. P. 72.

14. Zhang Y., Tang B., Luo J., Zhang J., Zhang Y., Tang B., Luo J., Zhang J. Deadline-Aware Dynamic Task Scheduling in Edge–Cloud Collaborative Computing. Electronics. 2022. Vol. 11, no. 15.

15. Li Z., Chang V., Ge J., Pan L., Hu H., Huang B. Energy-aware task offloading with deadline constraint in mobile edge computing. EURASIP Journal on Wireless Communications and Networking. 2021. Vol. 2021, no. 1. P. 56.

16. Nieto G., de la Iglesia I., Lopez-Novoa U., Perfecto C. Deep Reinforcement Learning techniques for dynamic task offloading in the 5G edge-cloud continuum. Journal of Cloud Computing. 2024. Vol. 13, no. 1. P. 94.

17. Taherdoost H., Madanchian M. Multi-Criteria Decision Making (MCDM) Methods and Concepts. Encyclopedia. 2023. Vol. 3. P. 77–87.

18. Dewi R. K., Hanggara B. T., Pinandito A. A Comparison Between AHP and Hybrid AHP for Mobile Based Culinary Recommendation System. International Journal of Interactive Mobile Technologies (iJIM). 2018. Vol. 12, no. 1. P. 133–140.

19. Lamrini L., Aberkane H., Abounaima M. C., Lamrini L. GPU-TOPSIS: A Complete Vectorized and Parallel Reformulation of the TOPSIS Method for Large-Scale Multi-Criteria Decision Making. Preprints, 2026.

20. Khan M. M., Imran Z., UL Hassan N. Performance Analysis of Double Auction Implementations for Peer-to-Peer Energy Trading on Resource Constrained Blockchain Platforms. 2025.

21. Kopanitsa Web3 Arduino: An Arduino (or ESP32) library to use web3 on Ethereum platform [Electronic resource]. URL: https://github.com/kopanitsa/web3-arduino (accessed 14.01.2025).

22. Ahmad A., Alabduljabbar A., Saad M., Nyang D., Kim J., Mohaisen D. Empirically comparing the performance of blockchain's consensus algorithms. IET Blockchain. 2021. Vol. 1, no. 1. P. 56–64.

23. Islam Md. M., Merlec M. M., In H. P. A Comparative Analysis of Proof-of-Authority Consensus Algorithms: Aura vs Clique. 2022.

Леонід ЧЕПЕЛЬ, Юрій БОЙКО
Київський національний університет імені Тараса Шевченка

БАГАТОФАКТОРНИЙ ГІБРИДНИЙ ПІДХІД ДО ПЕРИФЕРІЙНИХ ОБЧИСЛЕНЬ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЇ БЛОКЧЕЙН

Інтеграція пристроїв Інтернету речей (IoT) з периферійними обчисленнями (edge computing) підтримує застосунки у промисловості, охороні здоров'я та розумних містах. Периферійні обчислення переносять обробку даних ближче до джерела даних, зменшуючи залежність від центральної хмарної інфраструктури. Однак управління гетерогенними та географічно розподіленими системами створює виклики щодо координації, розповсюдження програмного забезпечення та встановлення довіри. Централізовані підходи створюють єдині точки відмови та обмежують масштабованість. Технологія блокчейн вирішує проблеми довіри та координації. Тим не менш, існуючим реалізаціям бракує внутрішньомережевої координації з метриками ресурсів у реальному часі, і вони не підтримують призначення задач з урахуванням навантаження.

Дана робота представляє гібридну платформу для IoT та периферійних обчислень на базі блокчейну. Платформа інтегрує багатофакторний вибір вузлів на основі репутації вузла та доступності ресурсів. Розроблений підхід включає управління таймаутами, контроль завантаженості черги задач, потокобезпечні паралельні транзакції та оркестрацію завдань на основі подій із виявленням застарілих даних. Позаланцюгові (off-chain) обчислення відокремлені від внутрішньоланцюгової (on-chain) блокчейн координації. Вибір вузла використовує функцію оцінки, яка поєднує рівень успішності з метриками ресурсів у реальному часі. Логіка координації реалізована через смарт-контракти на базі Ethereum, що забезпечує прозорість та незмінність даних. Перевірка системи була проведена в симульованому середовищі Docker.

Тестування показує, що вузли з вищими оцінками вибираються у всіх спостережуваних випадках. Однак, коли кілька завдань надсилаються в межах одного міжблокового інтервалу, оцінки вузлів не оновлюються. Ця проблема вирішується вибором вузлів на основі наступного блоку, а не поточного, та покращує надійність вибору. Платформа усуває прогалини, виявлені в існуючій літературі, і може бути використана у сценаріях промислових периферійних обчислень.

Ключові слова: периферійні обчислення, блокчейн, інтернет речей, комп'ютені системи, децентралізовані мережі