A trust model that ensures the correctness of computing in grid computing system

Pavlo Rehida[1,*,†], Oleg Savenko[1,†], Anatoliy Sachenko[2,3,†], Andriy Drozd[1,†], Petro Vizhevski[1,†]

[1] *Khmelnytskyi National University, Institutska str., 11, Khmelnytskyi, 29016, Ukraine*
[2] *Kazimierz Pulaski University of Technology and Humanities, Department of Informatics, Radom, Poland*
[3] *Research Institute for Intelligent Computer Systems, West Ukrainian National Unversity, Ternopil, Ukraine*

Abstract
The main objective of this article is to present model of a distributed computing system that ensures the correctness of computations performed with connected nodes. The article analyses modern approaches to designing systems and their functioning. It examines the advantages of chosen computing systems and the main challenges that need to be addressed for their successful application. The article describes a model that includes a central control unit and computing sub-systems, as well as the challenges associated with correct computing within this system. To address this problem, the article proposes utilizing the trust model approach, which classifies computing nodes by roles based on their level of trust. The functions of these roles are also discussed. Additionally, the article explores major approaches for designing the trust model, including the utilization of voting, behavioural analysis, and machine learning. Solutions for potential conflicts arising from voting are also presented. Special attention is paid to the process of gathering information and its further processing to form the behavioural portrait for each computing node in the system. Furthermore, the article covers data transformation and optimization to train models for predicting compromised nodes. Details of the performed experiment, including the used dataset, are also considered. The experiments performed show the dependence of prediction accuracy on the value of epochs set during model training.

Keywords
grid computing systems, behaviour analysis, binary classification

1. Introduction

In today's world the usage of distributed computing is popular across various aspects of modern life as for personal and for scientific purposes. These systems find applications in modelling complex physical processes, calculation of critical issues and risks, molecular modelling, processing Big Data for IoT systems, malware detection [1], training models for artificial

IntelITSIS'2024: 5th International Workshop on Intelligent Information Technologies and Systems of Information Security, March 27–29, 2024, Khmelnytskyi, Ukraine
* Corresponding author.
† These authors contributed equally.
pavlo.rehida@gmail.com (P. Rehida); savenko_oleg_st@ukr.net (O. Savenko) as@wunu.edu.ua (A. Sachenko); andriydrozdit@gmail.com (A. Drozd); petro.vizhevskyi@gmail.com (P. Vizhevski);
0000-0002-6591-7069 (P. Rehida); 0000-0002-4104-745X (O. Savenko); 0000-0002-0907-3682 (A. Sachenko); 0009-0008-1049-1911 (A. Drozd); 0009-0009-4851-0839 (P. Vizhevski);

intelligence[2], among other tasks. Distribution systems provide a substantial amount of computing resources by involving large number of independent computers to collectively solve tasks. Applying these systems to diverse tasks prompted the proposal of various types and organizational structures for distributed computing. However, these systems come with both advantages and drawbacks. Considering the huge utilization of these systems it is necessary to study its drawbacks and find new approaches to overcome their key issues.

The organization of the distributed computing requires solving a few important problems, that will allow to design efficient systems. These problems are task distribution, enabling fault-tolerant, data distribution, computing correctness checking, scalability, security [3,4] etc. This paper focuses on designing a model that guarantees the correctness of calculations based on a trust approach that is based on the behaviour aspects of computing elements through binary classification.

The first part of this paper discusses the analysis of modern approaches used to organize distributed systems. In covers cluster computing, grid computing, cloud computing and edge computing, along with their respective advantages, disadvantages, and applications. Subsequently, the main challenges of organizing dynamic distributed computing are considered. In following sections, the application of trust module aimed at ensuring the correctness of computation is presented. The analysis of the main approaches used to design this trust model is also discussed. It is proposed to utilize behavioural analysis to resolve conflict situations encountered during voting for the correctness of computation. The primary approach to analysing computing nodes involves collecting data to determine if a nide is compromised. The article outlines the dataset used and its operational process before being employed for model training. Lastly, the article presents the results of experiments conducted using TensorFlow for model training. The experiments showcase learning speed based on the specified number of records and epochs, along with the accuracy of predictions.

2. Modern approaches to organizing distributed computing

The diversity of tasks that were set before distributed computing has influenced the evolution of its various types, adapting them to specific requirements. On one hand, high-performance computing systems were designed to address computationally intensive tasks, commonly used for scientific simulations. On another hand, tasks that demand real-time processing and low latency led scientists to propose edge computing. Each of these approaches has its own advantages and specific organization. So, it is necessary to consider the most widely used types and their issues.

Cluster systems. This concept lies at the core of organizing distributed computing, aiming to attract substantial computing resources using the homogeneous elements. All these interconnected elements are configured to collectively provide a unified resource for individual tasks [5]. The elements of the system are located next to each other and utilizes special wired data transmission interfaces, enabling rapid transfer of large data volumes. These systems are easily scalable by adding new nodes. These additional nodes can swiftly replace those that have failed, allowing uninterrupted calculations to proceed. All these aspects determine such systems as leading in the processing of large and resource-intensive tasks. The planning and execution of accurate calculations is controlled by a central server. Virtualization is often used for efficient resource allocation. Cluster systems find applications across various domains, including scientific research modelling, software application servers, big data processing and database support.

Grid systems. This system can be characterized as one who involve computing elements for future computing, which can be located at long distances from each other [6]. Unlike cluster systems, various computing elements, both software and hardware, are often used here, so they are also called heterogeneous [10]. To organize communication and data exchange between these elements, traditional Internet connection technologies are most used. These systems have great potential for scalability. Thanks to different types of computing elements, it is possible to flexibly select the necessary resources for specific tasks, and also to provide a high level of reliability. Managing such systems is characterized by decentralization, that's means that they do not have a central control module. Computing elements operate with a degree of autonomy, allowing them to determine when and how much computing recourses to allocate for solving specific computational tasks. Due to the system's ability to involve a large number of elements, such systems can potentially accumulate significant computational power. This computation power can be used for different scientifical researches, for example in analysing potential malware software [7-9] A special case of such systems is volunteer computing, which involves the elements of all willing users on a voluntary basis. The dynamism of such systems sets a number of tasks before scientists that need to be solved in order to organize efficient and safe calculations. These tasks are: finding an effective algorithm of distributing tasks between computing elements, organizing the correctness of performed calculations, optimizing the overall performance of the system and it's elements, and finding efficient methods for data transmission inside the system.

Cloud systems. This type of systems refers to information technology solution whose primary purpose is to provide services such as: computational power, access to web-applications and different methods to store data. Cloud systems are characterized by their ability to offer services to users anywhere via the Internet. In this case users don't need to own any necessary hardware. From the user's point of view, cloud systems are easily scalable relative to the needs of their task. Cloud systems can be deployed with various access types, including: public, private, community and hybrid. In a public cloud, all services are accessible to registered users. While private cloud deployed within a company's infrastructure, providing resources exclusively for the needs of the owning company. This type ensures high security for services and data. Users are offered services through three scenarios: IaaS (Infrastructure as a Service – providing access to control operating systems, programs and network configurations), PaaS (Platform as a Service – includes databases and application development and deployment tools), SaaS (Software as a Service – provides resources to transfer the functionality of some applications to the browser, offering various advantages). To ensure the smooth and efficient operation of such system, critical tasks, while organizing it, include securing the system and implementing efficient algorithms for transferring large datasets.

Edge systems. Such type of system is quite new distributed system paradigm and it's appearance is connected with widespread use of the IoT devices in the modern world. The key aspect of this paradigm is rethinking where is data stored, processed, and managed for computation. The growing number of sensors increases accordingly the amount of information generated for specific purpose. Consequently, the time required for data transmission to a central node for processing, as well as the processing time itself, increases. In this concept, intermediate computational resources are proposed to perform partial or complete data processing before sending it to the central node. Such an approach offers an advantages in systems that need to operate in real-time, because needed data can be obtained by executing a single request addressing the control element in the corresponding area. Therefore, edge systems finds application in complex IoT systems, health and safety, autonomous vehicles, telecommunications systems, and energy sector. The main challenges for such systems include ensuring stable communication between network elements, securing data transmission and processing, and distributed control of system components.

3. Dynamic nature in grid computing systems

Grid systems are a concept that describes systems involving elements that can be located far away from each other. Also, grid systems define how elements inside it may participate, and by default, they have some level of autonomy. With this autonomy, elements can decide how long and to what extent they will perform necessary tasks for a given objective. Taking these characteristics into account, developers have utilized this concept, to implement *grid computing systems*. Basic model of such a system is presented in fig. 1.

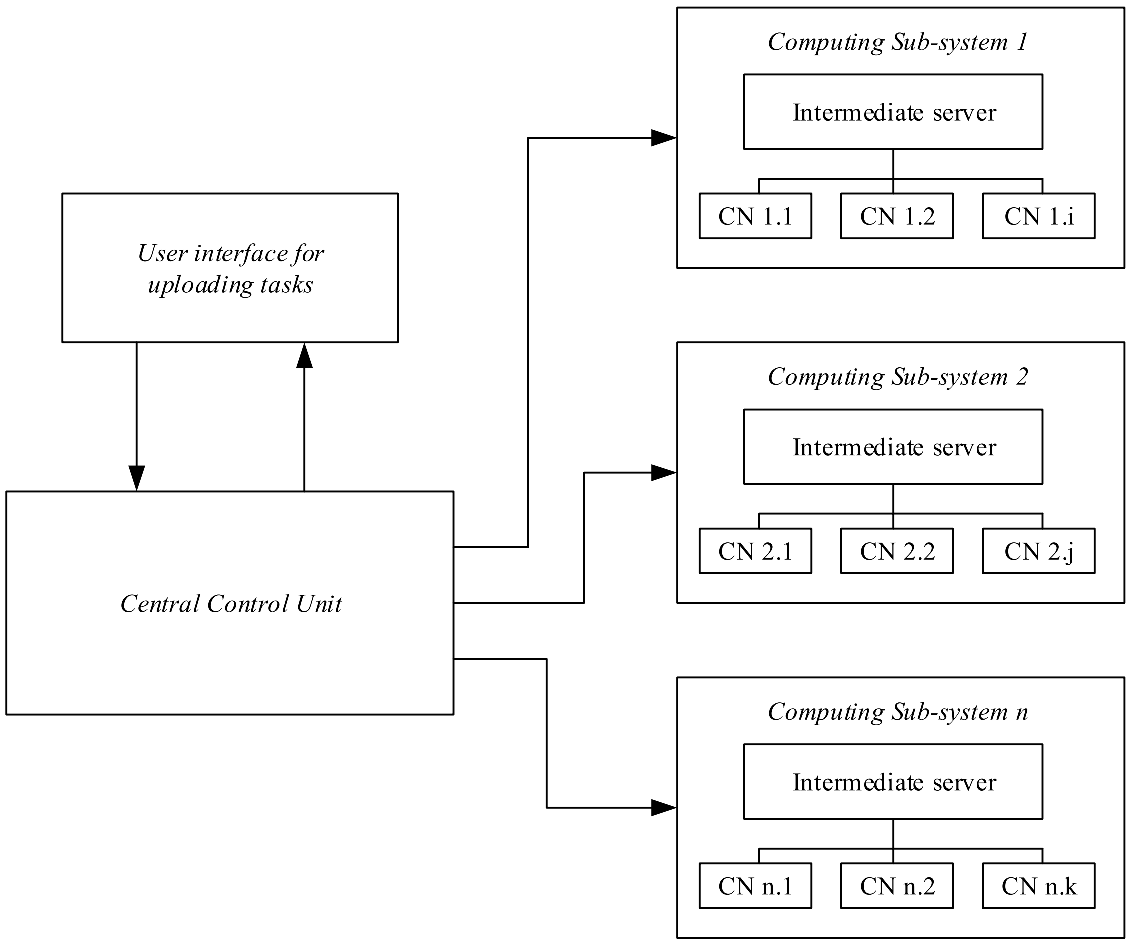


Figure 1. General model of grid computing systems.

In contrast to the traditional cluster computing system, grid computing presents challenges specific to its design. These challenges include the complexity of managing, security issues, interoperability challenges, network connectivity dependency, as well as load balancing and scheduling challenges. Additionally, scalability issues based on task and energy consumption. Most of these challenges are relevant to grid systems designed for various computation tasks. Despite these challenges, grid computing system are highly efficient and widely used in modern world of distributed computing. For example, there are a lot of successful systems that utilize the computational power of individuals who willingly contribute for scientific purposes on volunteer basis.

The most known systems are:

- *BOINC (Berkeley Open Infrastructure for Network Computing)* – software platform and is utilized by various universities worldwide. Scientific projects using this system cover fields such as mathematics, climate studies, astrophysics, computer science, and cryptography.
- *WCG (World Community Grid)* – global volunteer grid computing systems that also utilizes the computational power of users. Its primary research focus is in the field of health.
- *EGI (European Grid Initiative)* – defined as federation of computing and storage recourse providers. This platform focuses on several tasks, including the analysis of imaging data and the developing and common approach to implement digital twins.
- *CharityEngine* – this project based on BOINC technology. In contrast to participation for scientific purposes, volunteers provide resources for business needs. Business owners pay money, which is used for various charitable purposes.

Beside these projects involved into real-world applications, software enthusiasts collaborate into teams to propose alternative solutions. They create teams working on developing new grid computing applications based on different programming languages stacks. Most of these solutions are freely available and allows users to set up private grid computing system for their personal needs. The most known solutions are JPPF (task-based model of grid computing with dynamic node discovery feature based on Java), NGrid (.NET grid computing model with an efficient resource utilization approach) and Distri.js (grid computing system based on JavaScript that uses a browser as client to perform computation).

Existence of real-life grid computing systems and frameworks for personal grid system set up ensures that this type of computing is both popular and useful. These systems provide significant computational power and can be defined as *dynamic* computing system. The dynamic characteristic can be considered from various perspectives. From the viewpoint that these systems are designed for performing computations for different tasks, a key aspect is that the computing nodes are not belongs to system. Considering this fact, it is possible to define set of tasks that must be solved to ensure the efficient and secure computing:

- *Efficient resource utilization* – It is necessary to use the efficient algorithms for task distribution, taking into account that nodes connected to the system will differ from hardware viewpoint.

- *Providing scalability* – The system should be able to use additional resources from newly connected node efficiently.
- *Ensuring the fault-tolerance* – A high level of autonomy for each node allows them to stop performing computations at any time. The system must be prepared for this scenario.
- *Providing computation correctness* – The system should be prepared in any case node makes a mistake while computing its part of the task.
- *Network issue oriented tasks* – Communication in grid system mostly organized via the traditional Internet connections, so it is necessary to provide approaches that decrease the impact of network issues.

For the first task various, load-balancing algorithms and heuristics are commonly used. Such algorithms should take into account the type of the task and how well these tasks may be performed with parallel computation and the number of currently connected nodes in the system. These algorithms may also help in providing the scalability of the system if they are include a proactive approach. With proactive approach of the algorithm, it will build the next portion of computation, while nodes keep working on current one. In general, it will also minimize time of node idling. A high level of fault-tolerance may also be achieved by good task distribution between nodes and by including replication and redundancy approaches [11]. Computational correctness is achievable with the redundancy checks, which can be performed for the whole task or only for a part of it, deploying trust model or voting systems, etc. Network issues can be solved by implementing the specialized network protocols, data replications or even implementing SDN. In this case, SDN may be used for create an isolated network for some task with the an intermediate server, where the server and nodes will be close territorially to each other, so the network latency won't effect on the system's performance.

This paper is aimed to propose a modified trust model for grid computing system based on behavioural analysis of computational nodes with the inclusion of a binary classification tool.

4. Trust model approach

A trust model, in context of distributed computing system, refers to a software framework designed to manage trust relationships among system elements. In grid computing systems, trust is a critical factor that influences access control and collaboration between computing nodes. Considered how computing nodes participate in the grid system, it becomes evident that the central unit, responsible for all calculations, cannot trust to any of them. This limitation arises from the fact that computational server cannot exercise full control over any individual node due to the nature of distributed tasks performed by these nodes. The nature of task, in this context, implies that a single large task distributed across all elements depends on correctness of each individual node. Even a small part of the task calculated incorrectly can impact the overall result. Thus, on the one hand, the system cannot inherently trust any of the computing elements, and on the other hand, it must ensure the correct computation. A trust emerges as a viable solution to navigate this challenge.

Trust model are used in different areas, such as computer networks, IoT systems, social networks, multi-agent systems, healthcare systems and cloud computing. Trust models operate using various approaches, the most well-known are:

- *Reputation-based* – every computing node has its own reputation. Correctness of calculations may be achieved by considering the current level of reputation [12], which is based on successfully completed past tasks.
- *Voting-based* – voting, as a concept, is widely used in IoT systems [13] for making decisions based on collected data. Nodes in the system vote to establish a level of trust in processing events within the system.
- *Blockchain-based* – blockchain technology may be used in a trust model from different perspectives, including identification and secure data transmission. In distributed systems, it finds application [14] in resource management, security ensuring, network delays.
- *Game theory-based* – defines a set of rules for node how to collaborate and communicate with other nodes. It may be applied for establish secure message transmission between an intermediate processing unit and an edge element [15].
- *Behaviour-based* – this approach uses the concept of node's behavioural trust that based on analysis of past actions while processing tasks inside system [16]. This method is commonly used for cybersecurity in decentralised systems.
- *Machine learning-based* – trust systems also can utilize the AI concept that can be based on decision trees, K-nearest neighbour, naïve Bayes or random forests. Such trust model can be used as part of intrusion detection system [17]

Many modern trust models used in distributed systems do not rely solely on one of the aspects considered earlier. In most cases, they incorporate multiple aspects to achieve a higher level of efficiency. Developers take into account the main purpose of the system and the way if functions when designing the trust module for it.

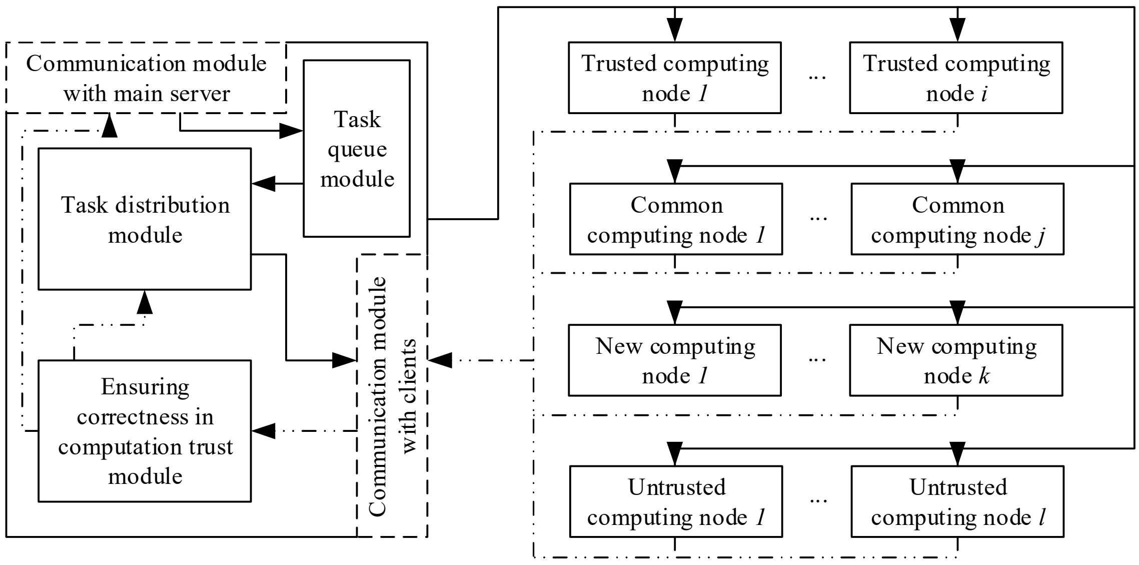
This paper introduces the concept of trust module for a grid computing system, with the primary goal of ensuring the correctness of computations. The proposed module based on combination of several approaches. The primary approach is based on the voting mechanisms, and to address potential conflicts during the voting, elements of machine learning and behavioural approaches are integrated. Conflicts that may arise during the voting are possible only if certain computing elements, responsible for computing the same tasks or even portion of them, return different results. In such cases, the system should utilize a rating system calculated based on the history of previous calculations for each element and its behaviour. It is proposed to access behaviour from two perspectives: node identification and node computation. Node identification refers to the physical information that can be obtained about host hardware and software via the client application. In most cases, users who wished to participate in computing do not frequently alter the characteristics of their hosts (including hardware and installed software). Considering this, when a user connects with uncommon characteristics, the system can identify this node as compromised. From the computational viewpoint, the system may verify the results of the calculation. Incorrect calculations may be associated with the presentence of malware on computing node. By combining these two approaches, the system can make a more accurate decision regarding the correctness of the computation. The system should also record successful calculations in terms of characteristics. This type of obtained data will be used to train the neural network.

5. Trust module for grid distributed system based on node behavioural analysis

For the proposed trust module, the system continuously obtains data from all connected computation nodes. Given that grid computing system are naturally designed to work with a large number of computation nodes, the main central control unit will have to create a set of computing groups, each with a leading intermediate server.

The whole grid computing system may be described as $GCS = \{ccu, css_1, \dots, css_n\}$, where ccu stands for *central control unit*, and css_n represents a *computing sub-system*. In context of communication, the ccu performs it with the intermediate server, which is part of css. Each css uses a trust module to ensure the correctness of performed computations and a set of computing nodes that differ in their usage perspective.

So, each sub-system can be defined as $CSS = \{is, tm, CE\}$, where is stands for intermediate server, tm for trust module, and CE is a set of different computing nodes. The model of such sub-system is shown on fig. 2.



Communication module with main server
Task queue module
Task distribution module
Ensuring correctness in computation trust module
Communication module with clients
Trusted computing node *1*
...
Trusted computing node *i*
Common computing node *1*
Common computing node *j*
New computing node *1*
New computing node *k*
Untrusted computing node *1*
Untrusted computing node *l*

Figure 2: Computing sub-system based with trust module

The trust system utilizes information about how different computing nodes perform computations, evaluate the returned result and gathers other relevant parameters, such as time taken to complete tasks, the correctness of task execution, and behavioural information. For each computing node, the trust system sets a trust value, which can be referred to as the *trust level of computing element.*

This value ranges from 0 to 1, and the trust module assigns different roles to each computing node, based on this value:

- *Trusted computing node* – system will never perform additional checks to the results that they have returned.
- *Common computing node* – system will never trust these elements, so it will send the parts of their tasks to the other computing nodes, to be sure that their calculations are correct.
- *New computing node* – these nodes start with some value of trust, and system will send them test tasks to evaluate their computation power and regular tasks to reached needed level of trust to become the common computing nodes.
- *Untrusted computing node* – node of this type made a lot of mistakes while computing tasks, that were defined while voting. System defines them as compromised and requires a lot of checks before they become as common computing node.

So, we can define $CN = \{tcn_1, \dots, tcn_i, ccn_1, \dots, ccn_j, ncn_1, \dots, ncn_k, ucn_1, \dots ucn_l\}$,
where tcn is a trusted computing node;
ccn – common computing node;
ncn – new computing node;
ucn – untrusted computing node.

As, it was described earlier, ccu should ensure that each css consists of no fewer than three trusted computing nodes. The system will assess the effectiveness of the computations performed and adjust the resulting rating for each computing node.

When a computing node completes its current task correctly, its rating will increase, while a computation with an incorrect result will decrease rating. It is suggested that the rating will change according to formula (1).

$$cnr_i = cnr_i + c * ck \tag{1}$$

where cn_i stands for current trust level of i computing node, c describes the correctness of performed computing, ck is a coefficient set by the administrator that influences how the level of trust will change after the computing task is completed.

It is suggested that the system administrator will set the basic values to help the system the role of each node, that takes part in task computing. For example, untrusted nodes have rating between 0 and 0.5, a common node has rating between 0.5 and 0.8, and a rating between 0.8 and 1 is assigned to trusted nodes. This rating will be updated based on the voting results, which are performed after each computing iteration in sub-system. To enable voting, the is in formed sub-system will distribute tasks among common computing nodes and include a portion of a task designed for another node. Depending on the load of trusted nodes, these portions may also be sent to them. After obtaining the results, the is will determine whether these tasks were computed correctly.

Working with nodes that have status ‘new’ in computing sub-system presents two challenges: the ccu lacks information about the computational power they can provide, making impossible to distribute tasks accurately, and it lacks information about the trust level of these nodes. It is proposed to use some test data to set up primary parameters for the new element, which may be equivalent to the least powerful client in the system, in terms of computing capability. Then, the system will adjust the portion of the task based on how quickly the previous task was computed. This evaluation is crucial for organizing the synchronous computing for all tasks. The formula for determining the trust level of new nodes may vary and can be adjusted by the administrator.

6. Behaviour analysis tool for resolving conflicts in trust module

Behaviour analysis is common approach that used in various fields of science for identification tasks of. In field of informational technology, it finds it application in several important tasks, including security monitoring to detect anomalies, analysing network traffic [18] to identify anomalous patterns, user authentication and authorization based on the analysis of typical behaviour patterns [19], optimization of distributed edge computing system [20], and predictive maintenance based on real-time equipment monitoring [21,22].

As mentioned earlier, the proposed trust model works with two sets of data to form the trust level of computing elements. The first set of data represents characteristics that can be collected using installed client node software, including information about the hardware and software being used.

The term “node software” refers to the application responsible for communicating with the sub-server and executing tasks assigned to the node. This application allows node to receive new tasks for computation. The information that can be obtained depends on the programming language used for its development. However, for most languages, it is possible to obtain the following information: *os* (operation system), *cpu_a* (CPU architecture), *cpu_m* (CPU model), *cpu_s* (CPU base speed), *cpu_n* (number of CPU cores), *ram* (total RAM), *home_dir* (home directory location on disk), *mac* (MAC address). Examples of this data are shown in Table 1.

Table 1
Node characteristics that describe software and hardware.

Computing node identifier	*os*	*cpu_a*	*cpu_m*	*cpu_s*	*cpu_n*	*ram*	*home_d*	*mac*
client1	win.	x64	Amd ryzen 5	3600	12	16	C://usr1	9c:b5:3c:9f:f1:4f
client1	lin.	x64	Intel core i5	3700	6	16	/home/usr2	0b:c0:21:87:58:57
client1	darv.	x64	Intel core i5	2000	8	16	/Users/usr3	78:7d:49:07:1d:b3

The second set of parameters describes how the computing client performs its computations for each task.

This data will include the following information: $task\_c$ (task complexity), $task\_p$ (time taken to perform the task), $task\_s$ (time required to send the task), $task\_r$ (time required to send results back), $task\_prod$(coefficient that describes the productivity of performed task). Examples of this data are shown in Table 2.

Table 2
Node characteristics that describe task computing.

Computing node identifier	*task_ c*	*task_p*	*task_s*	*task_r*	*task_prod*
client1	2000	6100	22	33	3.05
client1	1500	4400	19	32	2.93
client1	2000	6200	21	33	3.1

During operation, the intermediate server collects data for each task performed by every computing node. This data is proposed to be used for training a neural network, which will help to identify if a computing node was compromised. It was mentioned earlier that each subsystem requires at least three trusted nodes to ensure the correctness of computation. If two trusted nodes return the same results while a third one returns different result, the voting approach will resolve this situation. However, if all trusted nodes return different values, the system must decide which computation was correct. In such cases, the trained model will assist in determining if any of the trusted nodes were compromised.

7. Experiments

Before the collected data can be used for training the model, it is necessary to format it in a suitable way that can be utilized by most of the developed solutions for training. Each row in Table 1 and Table 2 represents the data characterizing one computed task on a client. Therefore, the system will generate one set of characteristics for each computed task on each computing node and store them until this data is used for training the personalized model. The data stored in the first table is not suitable for deep learning approaches because models do not support working with text information. To address this issue, it was decided to use one-hot encoding [23]. This approach was applied to the following characteristics: operating system, architecture, CPU model, home directory location and MAC address. However, one-hot encoding approach should be used correctly based on needs of the model being trained. If the system trains one model for all computing nodes, the text data should be classified and represented with IDs. In this case, it is necessary to use an algorithm to prepare the data, such as data modification approaches [24], which help retain critical data for learning and even select the informative features to better represent the patterns [25].

To train the model, it is also necessary to provide information about whether the computation was correct. This information can be obtained through voting. If the voting results are positive, the system will assign the value of 1; otherwise, it will assign a value of 0. By combining both of these datasets, it will be sufficient to train the model for binary classification. With the trained model, it will be possible to predict if the current node is compromised.

For this research, the scenario involves training the model for only one computing node. Our dataset for training includes 3000 records about its computation. To emulate a compromised node, a feature was added to the client application that deliberately changes its behaviour. This modification alters various characteristics, employing three major strategies: changing the hardware of the node, modifying the performance of computing, and adjusting the time for sending and receiving tasks and results, respectively. The first scenario describes a situation where a user account was stolen and someone attempts to compromise all calculations being performed by the sub-system. The second scenario may indicate that the node is infected with malware, and the third scenario refers to situations where the results may be distorted during transmission over the network. In general, this modifier had to create at least 20% of records indicating that the node was compromised. Subsequently, the obtained data was prepared for training the model. TensorFlow.js was chosen for training the model, as all client and server software is based on Node.js. Using this information, several models were generated with different numbers of training epochs.

After training models another set of records was defined that describes the node behaviour. This dataset was used to predict whether each node has normal or compromised activity inside system. Figure 3 illustrates the accuracy results of these models."

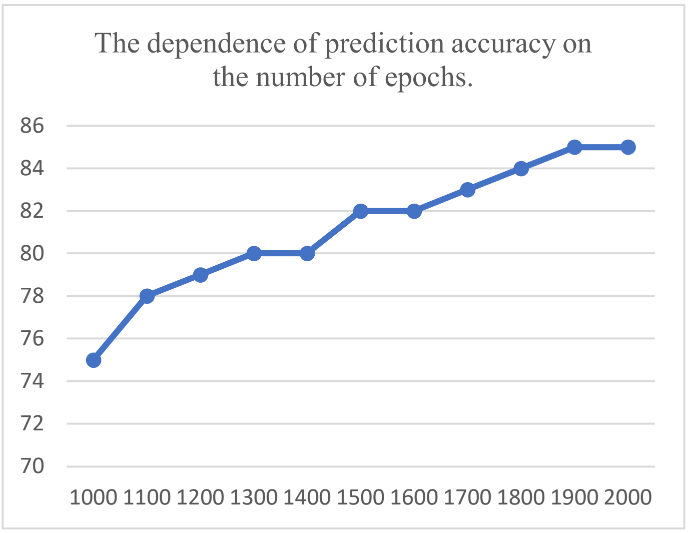


Figure 3: Graph that shows the accuracy of trained model depending on epochs

These results demonstrate that the number of epochs during model training influences prediction accuracy. Generally, as the number of epochs increases, the accuracy of the model improves. If we compare the results of training with 1000 epochs to achieve an additional 4% accuracy, it generally requires more than half of the additional time to train.

8. Conclusions

The main objective of this article is to present a trust model for grid computing systems that ensures the correctness of computation. After analysing the dynamic nature of grid computing, the primary challenges from the computing perspective were identified. Based on this, a modified trust module with behavioural analysis was proposed.

The results of experiments and the methodology used are presented graphically. While the obtained results shows a decent level of prediction accuracy, it may not be sufficient for safe distributed computing processes. Therefore, further research in this field is necessary. And the primary way for it is to adjust the data used to train the model. Since deep learning approach has shown better results in models trained on large datasets, it is important to focus on training a single module for all computing nodes. This will help to determine if the limited amount of data prevented creation of an accurate model.

Another way for future research involves to collecting more data for each node. However, collecting data for each node will require significantly more time, which may not be usable for newly launched systems.

References

[1] H. Kim, K. Lee, IIoT Malware Detection Using Edge Computing and Deep Learning for Cybersecurity in Smart Factories, Applied Sciences 12(15) (2022) 7679. doi: 10.3390/app12157679.

[2] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen, J.S. Rellermeyer, A survey on distributed machine learning, Acm computing surveys 53(2) (2020) 1-33. doi: 10.1145/3377454

[3] O. Pomorova, O. Savenko, S. Lysenko, A. Kryshchuk, A. Nicheporuk, A Technique for Detection of Bots Which Are Using Polymorphic Code, Computer Networks: 21st International Conference (2014) 265-276.

[4] A.F. Abate, A. Castiglione, L. Cimmino, D. De Angelis, S. Flauto, A. Volpe, On the (in)Security and Weaknesses of Commonly Used Applications on Large-Scale Distributed Systems, 24th International Conference On Control Systems And Computer Science (2023) 572-579. doi: 10.1109/CSCS59211.2023.00096

[5] H. Yviquel, M. Pereira, E. Francesquini, G. Valarini, G. Leite, P. Rosso, R. Ceccato, C. Cusihualpa, V. Dias, S. Rigo, A. Souza, The OpenMP cluster programming model, Workshop Proceedings of the 51st International Conference on Parallel Processing (2022) 1-11. doi: 10.1145/3547276.3548444

[6] M.K. Mishra, Y.S. Patel, M. Ghosh, G.B. Mund, A Review and Classification of Grid Computing Systems, International Journal of Computational Intelligence Research 13(3) (2017) 369-402.

[7] O. Savenko, S. Lysenko, A. Nicheporuk, B. Savenko, Approach for the Unknown Metamorphic Virus Detection, 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications 1 (2017) 71-76. doi: 10.1109/IDAACS.2017.8095052

[8] P. Rehida, O. Savenko, A. . Kashtalian, A. Sachenko, Malware Detection Tool Based on Emulator State Analysis, IEEE 12th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS) 1 (2023) 135-140. doi: 10.1109/IDAACS58523.2023.10348678

[9] G. Markowsky, O. Savenko, S. Lysenko, A. Nicheporuk, The Technique for Metamorphic Viruses' Detection Based on its Obfuscation Features Analysis, ICTERI workshops (2014) 680-687.

[10] K. Cao, Y. Liu, G. Meng, Q. Sun, An overview on edge computing research, IEEE access 8 (2020) 85714-85728. doi: 10.1109/ACCESS.2020.2991734

[11] M. Stetsiuk, A. Kashtalian. The methods of ensuring fault tolerance, survivability and protection of information of specialized information technologies under the influence of malicious software, Computer Systems and Information Technologies, 1 (2022), 36–44. doi: 10.31891/CSIT-2022-1-5

[12] M. Al-khafajiy, T. Baker, M. Asim, Z. Guo, R. Ranjan, A. Longo, D. Puthal, M. Tylor, COMITMENT: A fog computing trust management approach, Journal of Parallel and Distributed Computing (2020) 1-16. doi: 10.1016/j.jpdc.2019.10.006

[13] Y. Li, W. Susilo, G. Yang, Y. Yu, D. Liu, X. Du, M. Guizani, A blockchain-based self-tallying voting protocol in decentralized IoT, IEEE Transactions on Dependable and Secure Computing 19(1) (2020) 119-130. doi: 10.1109/TDSC.2020.2979856

[14] W. Li, J. Wu, J. Cao, N. Chan, Q. Zhang, R. Buyya, Blockchain-based trust management in cloud computing systems: a taxonomy, review and future directions, Journal of Cloud Computing, 10(1) (2021) 1-34. doi: 10.1186/s13677-021-00247-5

[15] C. Esposito, O. Tamburis, X. Su, C. Choi, Robust Decentralised Trust Management for the Internet of Things by Using Game Theory, Information Processing & Management, 57(6) (2020) 102308. doi: 10.1016/j.ipm.2020.102308

[16] P. Schmidt, F. Biessmann, T. Teubner. Transparency and trust in artificial intelligence systems, Journal of Decision Systems, 29(4) (2020) 1-19. doi: 10.1080/12460125.2020.1819094

[17] B. Mahbooba, R. Sahal, M. Serrano, W. Alosaimi, Trust in intrusion detection systems: An investigation of performance analysis for machine learning and deep learning models, Complexity (2021) 1-23. doi: 10.1155/2021/5538896

[18] M. Abbasi, A. Shahraki, A. Taherkordi, Deep Learning for Network Traffic Monitoring and Analysis (NTMA): A Survey, Computer Computation, 170 (2021) 19-41. doi: 10.1016/j.comcom.2021.01.021.

[19] H. Lu, Y. Zhang, Y. Li, C. Jiang, H. Abbas, User-Oriented Virtual Mobile Network Resource Management for Vehicle Communications, IEEE Transactions on Intelligent Transportation Systems, 22(6) (2020) 3521-3532. doi: 10.1109/TITS.2020.2991766

[20] P. Krishnan, S. Duttagupta, K. Achuthan, SDN/NFV security framework for fog-to-things computing infrastructure, Software: Practice and Experience 50(5) (2020) 757-800. doi: 10.1002/spe.2761

[21] O. Serradilla, E. Zugasti, J. Rodriguez, U. Zurutuza, Deep learning models for predictive maintenance: a survey, comparison, challenges and prospects, Applied Intelligence 52(10) (2022) 10934-10964. doi: 10.1007/s10489-021-03004-y

[22] A. Ucar, M. Karakose, N. Kırımça, Artificial Intelligence for Predictive Maintenance Applications: Key Components, Trustworthiness, and Future Trends, Applied Sciences 14(2) (2024) 898. doi: 10.3390/app14020898

[23] L. Yu, R. Rongtian, R. Chen, K.K. Lai, Missing Data Preprocessing in Credit Classification: One-Hot Encoding or Imputation?, Emerging Markets Finance and Trade 58(2) (2022) 472-482. doi: 10.1080/1540496X.2020.1825935

[24] P. Dhal, C. Azad, A comprehensive survey on feature selection in the various fields of machine learning, Applied Intelligence 52(4) (2022) 4543-4581. doi: 10.1007/s10489-021-02550-9

[25] S.R. Tiwari, K.K. Rana, Feature Selection in Big Data: Trends and Challenges, Data Science and Intelligent Applications: Proceedings of ICDSIA (2020) 83-98. doi: 10.1007/978-981-15-4474-3_9