UDC 667.021.1 DOI https://doi.org/10.35546/kntu2078-4481.2025.3.2.42

O. O. MAMCHYCH
Postgraduate Student at the Department of Electronic Computers
Kharkiv National University of Radioelectronics
ORCID: 0009-0001-6602-2929

M. O. VOLK
Doctor of Technical Science,
Professor at the Department of Electronic Computers
Kharkiv National University of Radioelectronics
ORCID: 0000-0003-4229-9904

MODEL AND METHODOLOGY OF MEASURING AND FORECASTING OF ENERGY IMPACT OF SECURITY RISKS MITIGATION APPROACHES IN DISTRIBUTED COMPUTING SYSTEMS ON PERSONAL MOBILE DEVICES

The rapid growth of cloud computing and artificial intelligence has drastically increased energy consumption in data centers. Volunteer computing using personal mobile devices represents a promising solution by potentially reducing energy demands. However, such systems face critical challenges related to reliability, security, and sustainability. The ***subject of this study*** *is the mitigation of three principal risks –* ***node outage, result cheating, and data fishing*** *– within mobile volunteer computing environments, and the evaluation of their* ***energy overhead****. The* ***goal of the research*** *is to extend a unified model capable of forecasting energy consumption under different security mitigation strategies and to quantify their trade-offs between efficiency and trustworthiness.* ***The methodology*** *combines mathematical modeling, simulation, and empirical measurements conducted on networks of mobile devices.* ***Case studies*** *employing ray tracing workloads were used to compare baseline energy efficiency with scenarios involving node outages, malicious result injection, and data leakage attempts.* ***Results*** *demonstrate that each mitigation strategy entails distinct energy implications. Node outage protection introduces moderate and predictable overheads that scale with failure probability. Cheating mitigation, based on redundancy and trust evaluation mechanisms, is identified as the most energy-expensive approach, with overhead rising exponentially in adversarial environments and, in extreme cases, collapsing system performance. Data fishing mitigation produces minimal additional energy consumption but restricts task allocation, potentially leading to incomplete job execution if trusted nodes are unavailable. The* ***scientific novelty*** *of this work lies in the integrated framework that unifies security and reliability risk mitigation with quantitative energy modeling in distributed computing. Unlike prior studies that address risks in isolation, this research systematically compares their energy impacts, establishing the taxonomy of overhead profiles and providing a foundation for balancing efficiency and security in future large-scale distributed systems.*

Key words: *volunteer computing, distributed computing, distributed computing systems, energy efficiency, information security, mathematical modeling.*

О. О. МАМЧИЧ
аспірант кафедри електронно-обчислювальних машин
Харківський національний університет радіоелектроніки
ORCID: 0009-0001-6602-2929

М. О. ВОЛК
доктор технічних наук,
професор кафедри електронно-обчислювальних машин
Харківський національний університет радіоелектроніки
ORCID: 0000-0003-4229-9904

МОДЕЛЬ ТА МЕТОДОЛОГІЯ ВИМІРЮВАННЯ ТА ПРОГНОЗУВАННЯ ВИТРАТ ЕНЕРГІЇ НА МІНІМІЗАЦІЮ РИЗИКІВ ІНФОРМАЦІЙНОЇ БЕЗПЕКИ В РОЗПОДІЛЕНИХ ОБЧИСЛЮВАЛЬНИХ СИСТЕМАХ НА ОСНОВІ ПЕСОНАЛЬНИХ МОБІЛЬНИХ ПРИСТРОЇВ

Стрімке зростання хмарних обчислень та штучного інтелекту суттєво збільшило енергоспоживання дата-центрів. Волонтерські обчислення (Volunterr Computing) з використанням персональних мобільних пристроїв є перспективним рішенням завдяки потенційному зниженню енергетичних витрат. Водночас такі системи

стикаються з критичними викликами, пов'язаними з надійністю, безпекою та стійкістю. ***Предметом даного дослідження*** *є мінімізація трьох ключових ризиків –* ***відмова вузлів, фальсифікація результатів та викрадення даних*** *– у середовищі мобільних добровільних обчислень, а також оцінка їх* ***енергетичних витрат****.* ***Мета роботи*** *полягає у розширенні єдиної моделі, здатної прогнозувати енергоспоживання за різних стратегій безпекового захисту, та кількісному визначенні компромісів між ефективністю і надійністю.* ***Методологія*** *дослідження поєднує математичне моделювання, імітаційні експерименти та емпіричні вимірювання на мережах мобільних пристроїв. У якості* ***прикладів*** *застосовувалися задачі трасування променів, що дозволило порівняти базовий рівень енергоефективності зі сценаріями відмов вузлів, навмисного спотворення результатів та спроб витоку даних. Отримані результати свідчать, що кожна стратегія захисту має власні енергетичні наслідки. Захист від відмов вузлів зумовлює помірні й передбачувані додаткові витрати, які зростають з ймовірністю відмови. Захист від фальсифікації результатів, заснований на механізмах надлишковості та оцінки довіри, є найбільш енерговитратним підходом, причому накладні витрати експоненційно зростають у ворожому середовищі й у крайніх випадках призводять до колапсу системи. Захист від викрадення даних створює мінімальні додаткові енергетичні витрати, проте обмежує розподіл задач, що може спричинити незавершеність обчислювальних робіт за відсутності довірених вузлів.* ***Наукова новизна*** *роботи полягає у створенні інтегрованої рамкової моделі, яка поєднує мінімізацію ризиків безпеки та надійності з кількісним енергетичним моделюванням у розподілених обчисленнях. На відміну від попередніх досліджень, що розглядали ризики ізольовано, це дослідження систематично порівнює їх енергетичний вплив, формує таксономію профілів накладних витрат і створює підґрунтя для збалансування ефективності та безпеки у майбутніх масштабних розподілених системах.*

Ключові слова: *волонтерські обчислення, розподілені обчислення, розподілені комп'ютерні системи, енергетична ефективність, інформаційна безпека, математичне моделювання.*

Introduction

Cloud computing demand has grown rapidly, with hyperscale and colocation data centers now dominating global workloads [1]. Between 2014 and 2023, U.S. data center electricity use nearly tripled, rising from ~60 TWh to 176 TWh (4.4 % of national consumption), largely driven by AI-accelerated servers. Projections for 2028 estimate 325–580 TWh, equal to 6.7–12 % of U.S. electricity demand. While efficiency improvements (e.g., PUE reduction, liquid cooling) mitigate some impact, the unprecedented rise of GPU-intensive training and inference workloads suggests that power use will continue outpacing efficiency gains, posing significant sustainability and infrastructure challenges.

As highlighted in the article The Case for Energy-Proportional Computing [2], current server infrastructures consume disproportionately high amounts of energy even at low utilization levels, where they typically operate. This inefficiency not only increases operational costs but also raises sustainability challenges. The concept of energy proportionality– designing systems that scale power usage in line with workload intensity – emerges as a critical paradigm. CPUs already demonstrate partial progress, but memory, storage, and networking subsystems remain major bottlenecks. Achieving true energy-proportional architectures would allow significant reductions in data center energy demand, potentially doubling efficiency without sacrificing performance. In the context of cloud computing's surging demand, addressing these inefficiencies is essential to balance performance, cost, and ecological responsibility.

As we can see, distributed computing and cloud computing is always related to a high energy consumption. Conventional stationary devices used in datacenters offer huge performance but at a high cost of energy. In our previous papers we suggested offloading of computing tasks to a distributed computing network built of personal mobile devices. This concept proven working and reducing power consumption by 3–4 times [3].

Using of personal mobile devices rises various security risks. There are approaches to mitigate those risks but they come at a cost of additional computational resources hence power. First, security risks represent a major concern. Personal devices are heterogeneous and usually lack uniform security standards. Malicious participants may intentionally inject corrupted data or manipulated results into the system (result cheating). Second, reliability and consistency risks arise from the variability of personal devices (node outage). Users may disconnect unpredictably, provide limited processing power, or operate under unstable network conditions. This volatility complicates scheduling and increases the probability of incomplete or inconsistent results. Third, users may want to steal sensitive data which may be a source data or results of computing (data fishing). Furthermore, diverse hardware and operating system configurations create reproducibility challenges, as energy and performance characteristics may differ substantially. The potential for "hidden exploitation" must be considered when designing fair participation models. Finally, sustainability risks are notable. Large-scale distributed computing may significantly increase overall energy consumption at the edge of the network. While individual contributions appear minor, aggregated impact can be substantial, undermining the intended efficiency gains if not managed with energy-aware scheduling. In summary, volunteer computing on personal devices offers opportunities but entails complex security, reliability, legal, and sustainability risks. Addressing these systematically is essential for safe and effective deployment. The main goal of this paper is to mitigate node outage, result cheating and data fishing risks in volunteer computing and compute how much of power will it cost.

State of the art

Volunteer computing (VC) offers a low-cost and sustainable model by aggregating idle resources of mobile devices, yet its openness and anonymity expose it to severe security risks. Android, the dominant VC platform, is increasingly targeted by advanced malware, threatening system reliability. To tackle this, the authors introduce MulDroid [4], a hybrid deep learning framework combining LSTM, CNN, and DNN to detect multi-vector Android malware. Using rigorous preprocessing of application manifests and evaluation on large-scale datasets (AMD, AndroZoo), MulDroid achieves 99.01 % detection accuracy and outperforms benchmark models in precision, recall, and F1-score, with only minor efficiency trade-offs. The work advances scalable and secure mobile VC by embedding autonomous malware defense at its core.

Volunteer computing (VC), and particularly its mobile variant (MVC), faces critical security challenges stemming from the heterogeneity and instability of volunteered resources. Protecting participant devices, ensuring trustworthy task execution, and safeguarding data transfers remain central obstacles. The authors address the problem of absent systematization in MVC research [5], where existing studies either overlook resource allocation or treat security and scheduling only superficially. To resolve this gap, they conduct a broad scoping review and develop the first comprehensive taxonomy of MVC resource allocation. Their approach categorizes methods by application type (independent, dependent, stream tasks), target resources (edge- or volunteer-oriented), and scheduling policy (objectives, constraints, algorithms). This structured framework not only clarifies current solutions but also identifies open issues such as incentive mechanisms, fairness, and integrated security models. The taxonomy thus provides a scientific foundation for advancing secure and efficient MVC platforms.

Unlike traditional cloud or edge servers, volunteer devices are unpredictable. These uncertainties create risks of task failures, delays, and security breaches, making providers hesitant to adopt VEC infrastructures. To tackle this problem, the authors propose VonEdgeSim [6], the first simulation framework specifically designed to mimic the dynamic and unreliable behavior of volunteer devices at the edge. The simulator models essential factors such as trust, availability, energy consumption, latency, and failures, offering a realistic environment for evaluating resource management strategies before real-world deployment. Implemented in Python for accessibility and extensibility, VonEdgeSim enables testing with IoT applications (e.g., augmented reality, infotainment, health monitoring), demonstrating significant reductions in task delay and execution time while improving fault tolerance. In conclusion, the authors close a critical research gap by delivering a practical, trust-aware simulation platform that empowers researchers to explore secure and efficient integration of volunteer resources in edge computing. Their results highlight the feasibility of incorporating volunteers into edge networks, paving the way for more reliable, scalable, and secure VEC infrastructures.

Volunteer computing provides massive, low-cost computational power but faces fundamental security and reliability challenges, as participating devices are anonymous, heterogeneous, and potentially malicious. The central problem lies in guaranteeing the correctness of results produced under such uncertain conditions. To address this, the authors introduce BOINC [7], an open-source middleware designed to ensure trustworthy high-throughput computing. Their solution integrates replication-based and adaptive result validation to detect faulty or cheating contributors, homogeneous redundancy and version control to counter platform inconsistencies, and sandboxing through restricted execution and virtualization to safeguard host systems. Together, these mechanisms establish a balance between security, efficiency, and scalability, transforming inherently insecure volunteer resources into a reliable infrastructure for large-scale scientific computation.

VFuse [8] suggests different approach, a browser-based decentralized architecture that mitigates security issues by leveraging WebAssembly for safe multi-language workflow execution, IPFS for distributed and tamper-resistant storage, and Ethereum blockchain with NFTs for transparent, verifiable, and incentivized participation. This approach solves the problem of secure orchestration and engagement in large-scale volunteer computing by combining P2P networking, blockchain-based trust, and user-friendly web notebooks into a robust framework. The main disadvantage of this system is high computational overhead and inability to utilize hardware acceleration in a flexible way.

Traditional volunteer computing suffers from unpredictable performance, availability, and security risks, making it unsuitable for sensitive or time-critical tasks. To overcome this, the authors of VECTrust [9] propose a two-stage probabilistic trust model that evaluates resources based on performance, agility, cost, and security (PACS). Using Dirichlet distributions, the model dynamically characterizes intra- and inter-cluster trust, enabling optimized resource allocation. Experimental validation on bioinformatics workflows demonstrates that VECTrust outperforms baseline algorithms, ensuring more secure, efficient, and balanced resource utilization across distributed volunteer environments.

We can state that there are various solutions mitigating security risks of volunteer computing. But none of those address all 3 highlighted problems (node outage, data fishing and result cheating). Plus, none of those problems investigate the cost of overhead while mitigating those risks. The goal of this paper is to cover all those 3 problems in terms of unified model and method for forecasting energy consumption in distributed computing systems based on stationary and mobile devices and calculate energy impact of each.

Case Study

As a base mathematical model, we will take a model from our previous research. As a computing Job we will take the same ray tracing engine but with motion blur effect: the first wave of Tasks is rendering of raw frames Artifacts, the second wave of Tasks will be blending of 4 raw frames into 1 blurred frame Artifact. This means that for Nth blurred frame

we need 4 raw frames: N, N-1, N-2 and N-3 (figure 1). Raw frames have Full HD resolution (1920 × 1080), rendered with 64 rays per pixel which gives us 133M rays per frame. Each ray trace takes average 260 float operation to trace: simple reflection, no rays multiplying, no more than 5 reflections, only spherical objects. Blurred frames have the same resolution, but blending of 4 pixels into one takes 24 float operations. The computing Job contains 2048 Tasks: 1024 raw frames renders and 1024 blended frames renders. The computing Job contains 3072 artifacts: 1024 source scene states, 1024 raw frames and 1024 blended frames. Source scene state is vector and has negligible size (up to 2kb), but each frame is 6.2Mb – we decided to keep those image as raw bitmaps to emphasize data transfer impact.

As a computing network we take a network of multiple Apple iPhone 12 mini devices connected into a network. For simplicity we will use CPU only computing. To simulate higher number of mobile devices we will run several execution contexts on each device. An execution context can execute a Task and has its own Scope of visible Artifacts, but several execution contexts cannot interact between each other. This allows graceful simulation of network transfer but does not affect the final amount of energy consumed on computation. For power consumption this time we will use a laboratory power source. Before case study we change mobile devices to 100 %, dim screens and disable all background activity to avoid any random irrelevant power drain. Other simulation parameters will also be taken from previous research as the device is the same: simulation tick duration – 1 second, network bandwidth 5.7 Mb per second via mobile internet, computing power – 16.4 GFLOP per second (63 million rays 260 FLOPs each), networking power consumption – 0.43 J per second, CPU consumption – 5.1 J per second.

Each measurement is conducted 3 times and averaged.

As a reference consumption we will take consumed power of computing on one single mobile device (to minimize data transfer) having 100 % uptime. This will be the minimal possible power can be consumed in terms of our model.

Simulation: time – 2,327 seconds, 11,747 J consumed.

Measurement: time – 2,468 seconds, 11,960 J consumed.

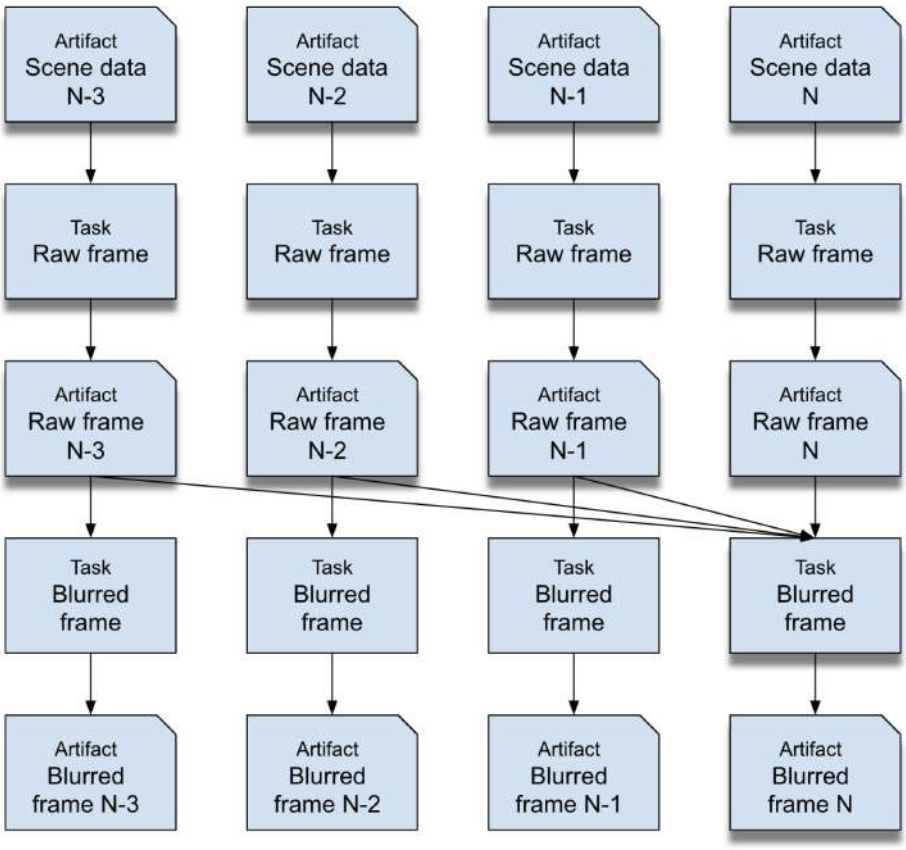

Fig. 1. Structure of a computing Job, relation between Tasks and Artifacts

Node outage power impact

The unified model already contains the simulation of nodes outage through Markov process: for each tick there is a probability to switch state from on to off and from off to on. When node goes off, it loses all the progress computing current task and all the Scope (known Artifacts). After node comes online it needs to download input Artifacts for any Task from scratch which will have an additional impact.

Let set a failover probability to 0.1 per average time of 1 raw frame Task calculation + 1 blurred frame task calculation duration, and restoring probability of 1 per tick. Average raw + blurred calculation is 2468/1024 = 2.41 seconds. If we assume normal distribution, the probability to fail within 1 tick (1 second) will be 0.043. Let setup these values and run the simulation.

Assume having 10 nodes.

Simulation: time – 243 seconds, 50 Tasks totally failed, 12263J consumed.

Measurement: time – 268 seconds, 47 Tasks totally failed, 12608J consumed.

Node cheat result mitigation

We investigate the case when some node can calculate result incorrectly. This can happen on purpose. For example, you have a computing task you want to offload to personal mobile devices and someone wants to compromise your results. They can add some compromised devices to the computing system, the system will distribute some tasks to those devices and the final result will be partially or completely damaged. To mitigate this the system can integrate some percentage of redundancy to the system to compare results of random nodes and detect cheaters. BOINC does this in their system, and our solution is inspired by theirs.

Our cheating protection subsystem is aimed at reduction of side effects caused by unscrupulous users and inappropriate hardware, yet reduce power consumption overhead as lower as possible. The subsystem has the following algorithm. Each node has following associated values: verification count (V) – number of sequentially validated results, errors counter (E) – number of errors results the node has submitted, excluded flag (X) – nodes with this flag are excluded from tasks distribution, trusted flag (T) – nodes with this flag considered trusted and should not be validated.

System has following global value: verification threshold (V_S) – minimal number of sequentially validated results the node should have to be considered reliable, ban threshold (E_S) – is node's error count goes above this threshold, node should be marked as excluded, forgiving rate R_F – rate of forgiving errors (decreasing node's E value) for reliable node after submission of valid results, unreliable node's redundancy level (R_U) is probability of result validation for unreliable node, reliable node redundancy level (R_R) is probability of result validation for reliable node, execution limit (L) – any Task can be evaluated no more than L times by non-trusted nodes. Usually, R_U is set to values close to 1 so near-every or every result of unreliable node is validated, but R_R is set to low values to avoid unnecessary redundancy for reliable nodes. Forgiving rate is introduced because the system itself may have flaws or Tasks may have their specifics, otherwise, some random rare algorithmic issue will lead to constant growth of E value indefinitely.

System tracks following audit trail for each executed Task and related Artifact: list of nodes evaluated or evaluating this Task. System can evaluate L(Task) as a number of nodes evaluated or evaluating this Task minus trusted nodes evaluated or evaluating this Task. Also, system tracks validation map: for each node it stores tasks it was validated on.

Algorithm is split into 2 phases: Task acquiring phase when system distributes particular task on a node and task submission phase when system receives Task from a node and takes decision. We also assume there are several thrust Task acquiring phase:

1. Validation dice roll. If X flag is set, reject Task acquiring, end phase.
2. If T flag is set no validation is required, provide any Task that is not evaluated or assigned to any node, end phase.
3. If $V >= V_S$, roll the dice with R_R rate. Otherwise roll the dice with R_U rate. If roll was positive, we decide to validate the result of Task node just acquired. If negative – provide any Task that is not evaluated or assigned to any node, end phase
4. Validation task selection. Prefer Task executed or being executed by trusted node with L(Task) < L. Otherwise prefer Task executed or being executed by at least 1 reliable node. Otherwise select any task assigned to other node. End phase.

Task submission phase – node evaluated Task and submitted Artifacts to system. If Task was not calculated by another node, do nothing. If task was calculated by multiple nodes, run the following node reliability test for each node.

1. If submitter node is trusted (T = true), add node-task pair to validation map, end of test.
2. If submitter node is excluded (X = true), end of test.
3. Compare submitted Task result to result from trusted node. If result matches, increment V, add node-task pair to validation map, end the test. If not matches, set V to 0, increment E.
4. Otherwise, compare submitted Task result to another node's result which validated is already validated for this Task. If result matches, increment V, add node-task pair to validation map, increment another node's V, end the test. If not matches, set V to 0, increment E.
5. Otherwise, compare submitted Task result to another node's result. If result matches, increment V, add node-task pair to validation map, end the test. If not matches, set V to 0, increment E.

After the test is finished positively, run forgiving routine: roll the dice with RF rate, if positive – decrease submitter node's E by 1, capped by 0.

After the test is finished negatively, of there is no validations for this Task in validation map, rollback the Task evaluation and Task assignment, consider it's never evaluated. If submitter node's $E > E_S$, exclude node by setting node's X to true.

Let's make the following configuration: reliability threshold $V_S = 5$, error threshold $E_S = 3$, forgiving rate $R_F = 0.1$, redundancy rate for unreliable node $R_U = 1$, redundancy rate for unreliable node $R_R = 0.05$.

Assume our computing system has 10 nodes: one is trusted (for example, tested system owner's device), 7 nodes will make correct calculations, one node will be doing only false results, one node will be doing 1 out of 4 false results.

Simulation: time – 311 seconds, 2 nodes excluded, 15,531 J consumed (+32 %).

Measurement: time – 336 seconds, 2 nodes excluded, 16,065 J consumed (+34 %).

Assume our computing system has 10 nodes: no trusted, 8 nodes will make correct calculations, one node will be doing only false results, one node will be doing 1 out of 4 false results.

Simulation: time – 311 seconds, 2 nodes excluded, 15,502 J consumed (+32 %).

Measurement: time – 334 seconds, 2 nodes excluded, 15,969 J consumed (+34 %).

Assume our computing system has 10 nodes: 1 trusted, 1 node will make correct calculations, 8 nodes will make false result every second time.

Simulation: time – 3,170 seconds, 9 nodes excluded, 48,500 J consumed (+312 %).

Measurement: time – 3,332 seconds, 9 nodes excluded, 51,991 J consumed (+334 %).

Assume our computing system has 10 nodes: no trusted, 2 nodes will make correct calculations, 8 nodes will make false result every second time.

Simulation: all nodes excluded.

Measurement: all nodes excluded.

Node data fishing mitigation

Malefactor may connect one or several compromised devices to our computing network which will result in stealing some source or intermediate data. Unfortunately, there are no reliable ways to prevent this at 100 % rate, but we can minimize amount of leaked data. To reach peak energy efficiency we should compute the whole Job on a single device to minimize data transfer efforts. But to minimize amount of leaked data we need to make the opposite: to involve as many devices as we can, so each computing device would "see" the least amount of data possible. And, if there are not enough devices to reach desired level of max leaked data, we need to involve trusted devices that are guaranteed to be reliable.

Each node will have an evaluation history containing all Task it's evaluated hence and all Artifacts seen. Also, node has trusted (T) flag.

System has a max number A_S of Artifacts that non-trusted node can "see".

Task acquiring flow will look as follows:

1. If node is trusted (T), approve acquiring, end flow.
2. Calculate number of Artifacts seen by node based on evaluation history of the node.
3. Pick available Task with biggest number of related (input + output) Artifacts that will not lead to exceeding A_S, taking into account Artifacts known to node. If there is such Task – allocate it and assign to the node. Otherwise – reject node.

If node is rejected, it will try to acquire tasks periodically since later unblocked Tasks can require less Artifacts and may depend on Artifacts known to the node.

Assume we have 10 nodes: 1 trusted, 9 regulars. System's $A_S = 400$ (> 10 %, we have 3,072 artifacts total).

Simulation: time – 237 seconds, average Scope is 312 Artifacts, max Scope is 322, 12,096 J consumed (+3 %).

Measurements: time – 252 seconds, average Scope is 312 Artifacts, max Scope is 320, 12,211 J consumed (+2 %).

Assumer we have 100 nodes: 10 trusted, 90 regulars. System's $A_S = 40$ (> 1 %).

Simulation: time – 25 seconds, average Scope is 31 Artifacts, max Scope is 36, 12,596 J consumed (+7.2 %).

Measurements: time – 28 seconds, average Scope is 31 Artifacts, max Scope is 40, 12875 J consumed (+7.6 %).

Assume we have 100 nodes: 1 trusted, 99 regulars. System's $A_S = 20$ (< 1 %).

Simulation: time – 856 seconds, average Scope is 33 Artifacts, max Scope is 2,590, 12,477 J consumed (+6 %).

Measurements: time – 912 seconds, average Scope is 34 Artifacts, max Scope is 2624, 12,938 J consumed (+8.1 %).

Experimental results discussion

As we can see every measure we take to mitigate security risks reflect in power consumption overhead. The simplest correlation with failover rate: higher failover rate results in re-distribution tasks into other nodes. In our case when 9 % tasks were failed, we've got about 5.4 % (4.4 %) of measured (simulated) power consumption overhead which look reasonable and not that high. Apparently, for high failure rates the overhead will grow to infinity as we close to 100 % fail rate.

On the other hand, cheating mitigation overhead is usually higher. If we have even 20 % of cheating nodes, we get 34 % (32 %) of measured (simulated) overhead. And in this case, it doesn't make sense if we have trusted nodes or not – the result is pretty the same, especially if total number of nodes is high. But when number of cheaters grows the overhead grows much faster. If we have trusted node (at least one) overhead can reach 334 % (312 %) of measured (simulated) overhead. If we don't have trusted node, the computing network collapses. We've missed the case when cheating nodes are interconnected and can coordinate their results and we don't have trusted nodes. This case requires additional investigation.

Data fishing mitigation does not generate such an impact by energy, neither by measured (2–7 %) nor by simulated (3–7 %). It doesn't differ from regular distribution of computational tasks into the maximum nodes possible if there are enough nodes to calculate the Job without exceeding data sharing threshold. And the overhead grows proportionally to

number of nodes. Even if we don't have enough nodes to not exceed data sharing threshold, we don't experience energy consumption impact because exceeding data is processed on trusted nodes. On the other hand, if we don't have enough of nodes and no trusted nodes, system will not be able to calculate the whole Job without exceeding data sharing threshold. At this point we can only rise the threshold or reconsider the Job execution.

We can see that every security risk mitigation has their own energy overhead profile: failure rate just adds a plain multiplier for low fail rates and blocks the computing network when fail rate approaches 100 %, cheater mitigation overhead is the highest even for no cheaters, and it grows to hundreds of percents when cheaters percentage is high. The data fishing mitigation energy impact is the lowest, but the data fishing mitigation results in outage if we don't have trusted nodes (figure 2).

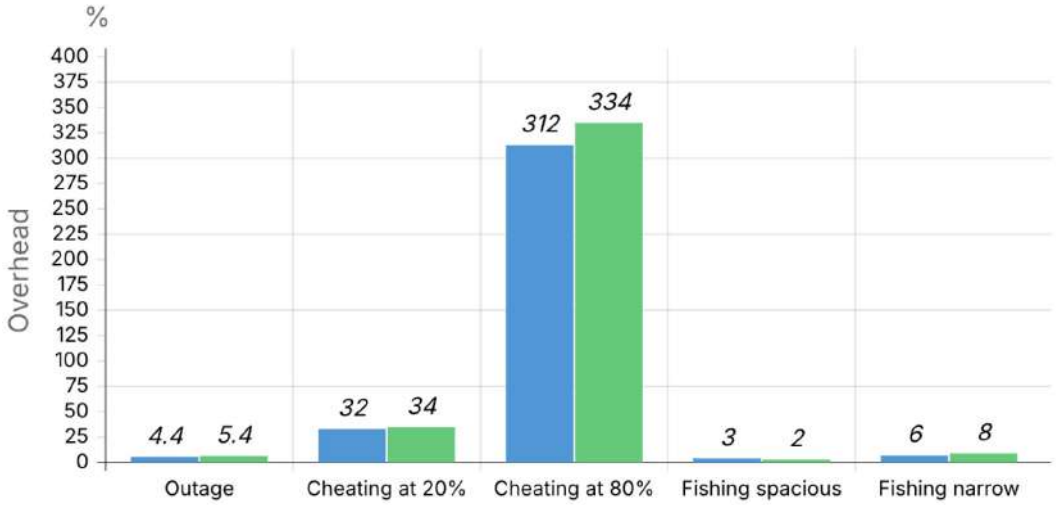

Fig. 2. Comparison of overheads from risks mitigation application

Conclusion

The conducted research highlights that integrating volunteer mobile devices into distributed computing infrastructures offers noticeable energy efficiency gains but inevitably introduces significant security and reliability risks. Through systematic modeling and experimentation, we demonstrated that mitigation strategies against node outage, cheating, and data fishing each carry distinct power consumption overheads, shaping the practical feasibility of such systems.

Node outage introduces moderate and predictable energy penalties, scaling with failure probability, while cheating mitigation emerges as the most energy-intensive, with overheads escalating rapidly under adversarial conditions and sometimes collapsing the system entirely. In contrast, data fishing mitigation imposes relatively low overhead but introduces operational constraints that may lead to task incompletion in absence of trusted nodes. Taken together, these results reveal a trade-off between energy efficiency and security in volunteer mobile computing. Effective deployment requires not only technical safeguards but also carefully balanced trust models and adaptive redundancy policies.

Future research will be targeted at several directions. First, development of better mitigation strategies, based of modern mathematical approaches. Second, we need a better model of energy losses for each mitigation strategy, we think there is a simplified "good enough" way of predicting those without complex simulation. In this research we have assumed that trust nodes are the same mobile devices, which resulted in execution time bottleneck. For real usage, to keep execution time under control, it would be better to use conventional data center hardware as a fallback. So, third target is to investigate mitigation strategies taking into account different energy consumption of trusted nodes to minimize energy profile, as well as growth of power consumption when risks mitigation strategies stick to fallback trusted conventional hardware.

Bibliography

1. Shehabi, A., Newkirk, A, Smith, S.J., Hubbard, A., Lei, N., Siddik, Md A. B., Holecek, B., Koomey, J., Masanet, E., & Sartor, D. 2024 United States Data Center Energy Usage Report Lawrence Berkley National Laboratory, 2024, Report LBNL-2001637, DOI: 10.71468/P1WC7Q

2. Barroso, L. A., Hölzle, U., The Case for Energy-Proportional Computing, 2007, Computer, no. 40, vol. 12, pp. 33–37. DOI:10.1109/mc.2007.443

3. Mamchych O., Volk M., A unified model and method for forecasting energy consumption in distributed computing systems based on stationary and mobile devices, RADIOELECTRONIC AND COMPUTER SYSTEMS, 2024, vol. 2, pp. 120–135, DOI: 10.32620/reks.2024.2.10

4. Bibi, I., Akhunzada, A., Malik, J., Khan, M. K., & Dawood, M. Secure Distributed Mobile Volunteer Computing with Android, ACM Transactions on Internet Technology, 2021, no. 22, vol. 1, pp. 1–21. DOI:10.1145/3428151

5. Ma, P., Garg, S. & Barika, M. Research allocation in mobile volunteer computing system: Taxonomy, challenges and future work, Future Generation Computer Systems, 2024, no. 154, pp. 251–265. DOI:10.1016/j.future.2024.01.015

6. Alsenani, Y. VonEdgeSim: A Framework for Simulating IoT Application in Volunteer Edge Computing, Electronics, 2024, no. 13(20), p. 4124. DOI:10.3390/electronics13204124

7. Anderson, D. P. BOINC: A Platform for Volunteer Computing, Journal of Grid Computing, 2019, no. 18(1), pp. 99–122. DOI:10.1007/s10723-019-09497-9

8. Antelmi, A., D'Ambrosio, G, Petta, A., Serra, L., & Spagnuolo, C. A Volunteer Computing Architecture for Computational Workflows on Decentralized Web, IEEE Access, 2022, no. 10, pp. 98993–99010. DOI:10.1109/access.2022.3207167

9. Pandey, A., Calyam, P., Debroy, S., Wang, S., & Alarcon, M.L. VEC Trust Proceedings of the 14th IEEE/ACM International Conference on Utility and Cloud Computing: 2021 IEEE/ACM 14th International Conference on Utility and Cloud Computing, 2021, doi:10.1145/3468737.3494099

Дата першого надходження рукопису до видання: 28.09.2025
Дата прийнятого до друку рукопису після рецензування: 24.10.2025
Дата публікації: 28.11.2025