

ПРОЄКТУВАННЯ СИСТЕМИ ЗАХИЩЕНОГО ВІДДАЛЕНОГО ДОСТУПУ ДО GPU-РЕСУРСІВ ІЗ МОНІТОРИНГОМ ТА АНАЛІЗОМ ЯКОСТІ МЕРЕЖЕВОГО З'ЄДНАННЯ

Кравченко Кирило Андрійович

студент групи МППм-1-26-1.4 д.,

*Київський столичний університет імені Бориса Грінченка, м. Київ,
науковий керівник — к.т.н., доц.Машкіна І.В.*

ВСТУП

Актуальність і постановка проблеми. Графічні процесори (GPU) є базовим обчислювальним ресурсом для навчання нейронних мереж, оброблення зображень і паралельних обчислень. Спільне використання GPU-сервера підвищує доступність обчислювальних ресурсів, однак віддалений доступ одночасно створює ризики компрометації облікових даних, несанкціонованого виконання коду, витоку дослідницьких матеріалів і виснаження ресурсів. Принципи Zero Trust передбачають відсутність неявної довіри лише на підставі мережевого розташування та потребують окремої автентифікації й авторизації доступу до ресурсу [1, 2].

Додаткова проблема полягає у визначенні причин низької якості віддаленої роботи. Зростання затримки, втрати пакетів, обмеження пропускної здатності, перевантаження GPU, CPU або сховища та очікування в черзі можуть мати подібні зовнішні прояви. Тому перевірку захищеності доступу доцільно поєднати з корельованим спостереженням за мережевим каналом, обчислювальним вузлом і прикладним сервісом. Дослідження GPU API remoting також демонструють суттєву залежність продуктивності від мережевих характеристик і конкретного типу навантаження [3].

Мета дослідження — спроектувати систему захищеного віддаленого доступу до GPU-ресурсів, у якій контроль ідентичності та прав користувачів поєднано з моніторингом і аналізом якості мережевого з'єднання для розмежування мережевих, ресурсних і прикладних причин деградації сервісу.

Об'єкт дослідження — процес захищеного віддаленого використання спільних GPU-ресурсів у мережевому середовищі.

Предмет дослідження — моделі, архітектурні рішення, програмні засоби та показники захищеного доступу, моніторингу і діагностування якості віддаленої роботи з GPU-ресурсами.

Для досягнення мети поставлено такі завдання: 1) проаналізувати загрози та вимоги до захищеного віддаленого доступу; 2) обґрунтувати архітектуру системи на основі сегментації, тунелювання, багатофакторної автентифікації та рольового контролю; 3) визначити набір мережевих, ресурсних і прикладних метрик; 4) розробити правила первинного діагностування причин деградації; 5) сформулювати методику експериментальної перевірки архітектури, продуктивності та коректності діагностичних правил.

АНАЛІЗ ДОСЛІДЖЕНЬ І ЗАСОБІВ

NIST SP 800-207 визначає Zero Trust Architecture як підхід, за якого довіра не надається автоматично через фізичне або мережеве розташування, а автентифікація та авторизація виконуються до встановлення сесії з ресурсом [1]. Практичний NIST SP 1800-35 деталізує реалізацію Zero Trust, зокрема керування ідентичностями та мікросегментацію [2]. Для захищеного тунелювання обрано WireGuard, який пов'язує відкритий ключ вузла з дозволеними тунельними адресами [4]. Водночас тунельна автентифікація вузла не замінює ідентифікацію користувача, тому вона має доповнюватися прикладною автентифікацією, авторизацією та процедурою відкликання доступу.

Для спостереження за GPU доцільно застосувати NVIDIA DCGM Exporter, який експортує телеметрію DCGM у форматі Prometheus [5]. Prometheus зберігає метрики як часові ряди з мітками та придатний для зіставлення показників різних компонентів [6]. Jupyter Server потребує особливої уваги до контролю доступу, оскільки доступ до сервера фактично надає можливість виконувати код; у сучасних версіях автентифікація та авторизація розділені як окремі етапи [7]. Для активних мережевих вимірювань використовується iperf3 [8], а для відтворюваної емуляції затримок і втрат — Linux netem [10].

МЕТОДИ ТА АРХІТЕКТУРА СИСТЕМИ

Дослідження має проектно-експериментальний характер. Використано аналіз вимог і загроз, функціональну декомпозицію, порівняння засобів захищеного доступу, сценарний аналіз відмов та планування контрольованого експерименту. Розглядаються три типи навантаження: інтерактивна робота через SSH/Jupyter, передавання даних і тривалі GPU-обчислення. Для них застосовуються різні критерії якості, оскільки чутливість до затримки та пропускної здатності відрізняється.

Модель загроз охоплює перехоплення трафіку, компрометацію ключа або токена, доступ до чужих файлів, надмірне використання GPU та відкриття служб моніторингу стороннім особам [1, 2, 7]. Передбачено індивідуальні ключі WireGuard, фільтрацію між сегментами, MFA у сервісі керування доступом, рольові дозволи та окремі користувацькі середовища. Контейнери запускаються без привілейованого режиму, з виділеними каталогами та обмеженнями CPU/RAM; при виконанні взаємно недовіреного коду контейнеризація не розглядається як достатня межа ізоляції.

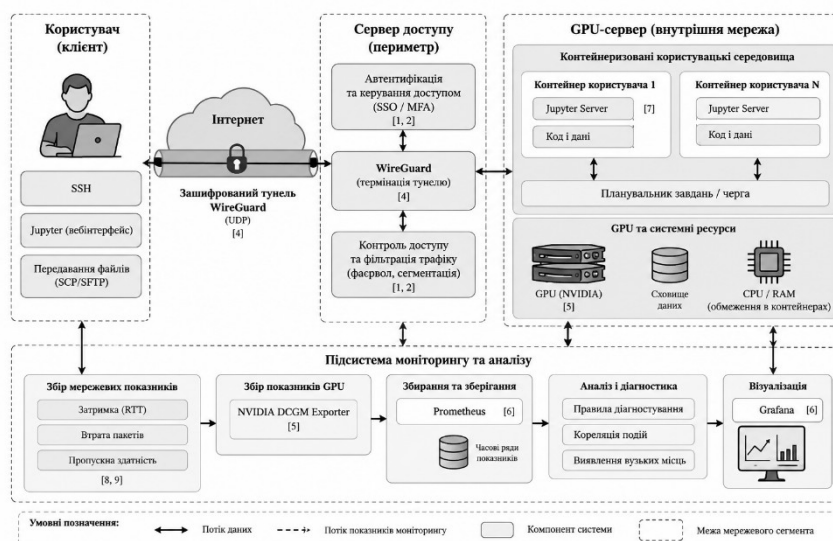


Рис. 1. Узагальнена архітектура системи

Підсистема спостереження логічно охоплює всі рівні: клієнтські проби характеризують якість каналу через той самий тунель, яким працює користувач; шлюз надає мережеві та системні показники; GPU-вузол — телеметрію DCGM, CPU/RAM, стан черги та прикладні події [5]. Дані агрегуються в Prometheus і візуалізуються на спільній часовій шкалі [6]. Такий підхід дає змогу використовувати часовий збіг змін як підставу для перевірки гіпотези, але не як автоматичний доказ причинності.

ПОКАЗНИКИ ТА ПРАВИЛА ДІАГНОСТУВАННЯ

Для мережевого каналу планується визначитись медіана та 95-й перцентиль RTT, частка невдалих прикладних проб, TCP-пропускна здатність, UDP-втрати та джитер; активні TCP/UDP-вимірювання можуть виконуватись засобами `iperf3` [8]. Варіація RTT між послідовними пробами буде використана лише як діагностична ознака і не ототожнюється з односторонньою IP Packet Delay Variation, стандартизованою RFC 3393 [9]. Для GPU планується контроль завантаження, використання відеопам'яті, температура та потужність; додатково — CPU, RAM, дискові операції, стан черги, час входу, запуск середовища і розриви сесій.

Стан з'єднання доцільно класифікувати як «прийнятний», «погіршений», «недоступний» або «невідомий». Порогові значення не слід задавати апріорно: їх потрібно калібрувати окремо для інтерактивної роботи та передавання даних за результатами базових вимірювань. Для зменшення частих перемикань статусу варто використовувати підтвердження стану в кількох послідовних часових вікнах і різні умови переходу до погіршення та відновлення.

Приклади правил первинного діагностування

Ознаки	Робоча гіпотеза	Перевірка
RTT/невдалі проби зростають; GPU не перевантажений	Проблема каналу або тунелю	Порівняти проби до шлюзу й сервісу; втрати; CPU шлюзу
Мережа стабільна; GPU зайнятий; черга зростає	Дефіцит GPU-ресурсів	Перевірити активні завдання та час очікування
Передавання файлів повільне; RTT стабільний	Обмеження каналу або сховища	Зіставити TCP-тест, диск і навантаження шлюзу
Тунель доступний; прикладний запит неуспішний	Відмова сервісу/авторизації	Перевірити службу, HTTP/SSH-відповідь і журнал доступу
Телеметрія відсутня або застаріла	Недостатність спостережень	Стан «невідомо»; перевірити exporter і канал збору

МЕТОДИКА ЕКСПЕРИМЕНТАЛЬНОГО ДОСЛІДЖЕННЯ

Дослідження передбачає реалізацію прототипу та контрольований експеримент. Базовий сценарій — порівняння захищеного прикладного доступу в довірєній лабораторній мережі з доступом через WireGuard [4] за однакових сервера, клієнта, набору даних і навантаження. За допомогою netem послідовно моделюються затримка, втрата пакетів та обмеження швидкості [10]. Мережеві фактори спочатку змінюються окремо від GPU/CPU/дискового навантаження, після чого досліджуються комбіновані сценарії.

У кожному сценарії доцільно виконувати серію повторів із фіксованими інтервалами прогрівання та вимірювання. Порівнюються час встановлення доступу, час відповіді сервісу, передавання контрольного файлу, медіана та 95-й перцентиль RTT, пропускна здатність і використання ресурсів. Окремо перевіряються відхилення неавторизованого доступу, відкликання ключа/облікового запису, завершення активної сесії та збереження обчислювального завдання після розриву мережі. Якість діагностичних правил оцінюється за часткою правильних визначень причини, часом виявлення та кількістю хибних сповіщень.

Завдання подальшого дослідження: 1) реалізувати експериментальний стенд і прототип підсистем доступу та моніторингу; 2) сформувати базові профілі мережевої та ресурсної продуктивності; 3) експериментально визначити пороги станів для різних типів навантаження; 4) **оцінити** накладні витрати захищеного тунелю та моніторингу; 5) перевірити точність правил діагностування на ізольованих і комбінованих відмовах; 6) оцінити стійкість механізмів відкликання доступу й ізоляції користувацьких середовищ.

ВИСНОВКИ

У роботі буде запропоновано архітектуру захищеного віддаленого доступу до GPU-ресурсів, що має поєднувати шифрований тунель [4], незалежну автентифікацію та авторизацію користувача відповідно до принципів Zero Trust [1, 2], ізоляцію робочих середовищ і багаторівневий моніторинг GPU та мережевих характеристик [5, 6, 8, 11]. На відміну від підходу, що контролює лише доступність сервера або лише завантаження GPU, запропоновано спільно аналізувати мережеві, ресурсні та прикладні показники. Сформовані правила будуть мати характер проєктних гіпотез; їх ефективність і порогові значення потребують експериментального підтвердження. Практичним результатом має стати прототип, здатний не лише фіксувати деградацію, а й обґрунтовано розрізняти її ймовірні мережеві, ресурсні та прикладні причини.

ДЖЕРЕЛА

1. Rose S., Borchert O., Mitchell S., Connelly S. Zero Trust Architecture. NIST SP 800-207. 2020. DOI: <https://doi.org/10.6028/NIST.SP.800-207>.
2. Kerman A. et al. Implementing a Zero Trust Architecture: High-Level Document. NIST SP 1800-35. 2025. DOI: <https://doi.org/10.6028/NIST.SP.1800-35>.
3. Wang T., Chen Z., Wei X., Gu J., Chen R., Chen H. Characterizing Network Requirements for GPU API Remoting in AI Applications. arXiv:2401.13354. 2024. DOI: <https://doi.org/10.48550/arXiv.2401.13354>.
4. Donenfeld J. A. WireGuard: Next Generation Kernel Network Tunnel. Draft revision e2da747. 2020. URL: <https://www.wireguard.com/papers/wireguard.pdf>.
5. NVIDIA. DCGM Exporter Documentation. URL: <https://docs.nvidia.com/datacenter/dcgm/latest/reference/command-line-reference/dcgm-exporter.html>.
6. Prometheus. Overview. URL: <https://prometheus.io/docs/introduction/overview/>.
7. Jupyter Server. Security in the Jupyter Server. URL: <https://jupyter-server.readthedocs.io/en/latest/operators/security.html>.
8. ESnet. iperf3 Documentation. URL: <https://software.es.net/iperf/>.
9. Demichelis C., Chimento P. IP Packet Delay Variation Metric for IP Performance Metrics (IPPM). RFC 3393. 2002. DOI: <https://doi.org/10.17487/RFC3393>.
10. Hemminger S. tc-netem(8) — Network Emulator. Linux manual page. URL: <https://man7.org/linux/man-pages/man8/tc-netem.8.html>.
11. Abramov, V., Astafieva, M., Boiko, M., Bodnenko, D., Bushma, A., Vember, V., Hlushak, O., Zhylytsov, O., Ilich, L., Kobets, N., Kovaliuk, T., Kuchakovska, H., Lytvyn, O., Lytvyn, P., Mashkina, I., Morze, N., Nosenko, T., Proshkin, V., Radchenko, S., & Yaskevych, V. (2021). Theoretical and practical aspects of the use of mathematical methods and information technology in education and science. URL: <https://doi.org/10.28925/9720213284km>