УДК 004.75:004.932.72'1
DOI https://doi.org/10.32782/2663-5941/2026.1.2/47

Фоменко В.О.
https://orcid.org/0009-0000-1890-5682
Сумський державний університет

МЕТОДИ ТУМАННИХ ОБЧИСЛЕНЬ (FOG COMPUTING) ДЛЯ ЗМЕНШЕННЯ ЗАТРИМКИ ПЕРЕДАЧІ ДАНИХ У СИСТЕМАХ РОЗПІЗНАВАННЯ ОБЛИЧ

Робота присвячена дослідженню методів fog computing для зменшення затримки передачі даних у системах розпізнавання облич на основі IoT-камер. Проведено експериментальне порівняння fog computing та традиційного cloud computing підходів шляхом симуляції обробки 100 задач розпізнавання облич, розподілених між 5 fog-вузлами та централізованим хмарним сервером. Fog-вузли розміщені рівномірно у просторі 100×100 м² з обчислювальною потужністю від 100 до 200 GFLOPS та ємністю пам'яті 50-100 ГБ. Хмарний сервер має потужність 500 GFLOPS але базову мережеву латентність 150 мс. Задачі розпізнавання генеруються з розміром зображень 0,5-3 МБ та обчислювальною складністю 5-20 GFLOPS. Розроблено scheduler-алгоритм на основі Particle Swarm Optimization для оптимального розподілу задач між fog-вузлами з урахуванням географічної близькості, обчислювальної потужності та поточного завантаження. Експериментальні результати демонструють зменшення середньої затримки обробки на 48,69% (з 174,77 мс до 89,68 мс), підвищення пропускної здатності у 1,94 рази (з 5,72 задач/с до 11,15 задач/с) та зниження споживання енергії на 67,9% (з 139,82 у.о. до 44,84 у.о.). Всі 100 задач успішно оброблені локально на fog-вузлах без необхідності відвантаження до хмари. Розподіл навантаження показує, що fog-вузли обробили від 7 до 32 задач залежно від обчислювальної потужності та географічного розташування. Fog Node 3 з потужністю 152,97 GFLOPS обробив найбільше задач (32) з використанням ресурсів 81,02%, тоді як Fog Node 4 з потужністю 157,20 GFLOPS обробив лише 7 задач через менш вигідне розташування. Медіанна затримка становить 77,31 мс для fog computing проти 175,53 мс для cloud computing. Мінімальна затримка у fog-архітектурі досягає 27,23 мс, тоді як у cloud-архітектурі навіть найшвидша обробка вимагає мінімум 160,28 мс через базову мережеву латентність. Fog computing забезпечує вищу якість обслуговування з відсотком SLA виконання 98,5% проти 92,3% для cloud computing та коефіцієнтом надійності 0,97 проти 0,94 відповідно.

Ключові слова: *fog computing, розпізнавання облич, затримка передачі даних, PSO оптимізація, IoT камери, edge computing.*

Постановка проблеми. Системи розпізнавання облич у розподілених IoT-середовищах стикаються з проблемою затримок. Основне джерело затримки – передача даних у хмару. За cloud-орієнтованої архітектури великі відеопотоки потрібно передавати назовні. Дані йдуть через Інтернет у датацентри. Через це типова затримка становить 150–300 мс. Для критичних сценаріїв такий рівень латентності неприйнятний. Це стосується безпеки аеропортів, систем контролю доступу та міського відеомоніторингу.

За результатами симуляції cloud computing має середню затримку 174,77 мс. Мінімальна затримка становить 160,28 мс. Це фіксується навіть для простих задач. Fog computing пропонує проміжний рівень обчислень. Ресурси розміщуються між IoT-пристроями та хмарою. Однак така парадигма вимагає ефективного розподілу навантаження.

Проблема зводиться до вибору стратегії розподілу задач розпізнавання. Розподіл виконується між fog-вузлами системи. Потрібно враховувати географічну близькість камер до вузлів. Також важлива обчислювальна потужність кожного вузла. Третій фактор – поточне завантаження системи. Без ефективного scheduler-алгоритму fog-архітектура працює нестабільно. Можливе перевантаження частини вузлів. Інші вузли при цьому простоюють. Це нівелює переваги розподіленої обробки.

Аналіз останніх досліджень і публікацій. Сахамі М. Е. [1] запропонував ефективний метод розподілу ресурсів в IoT-пристроях на основі fog computing для задач детекції облич, досліджуючи алгоритми динамічного розподілу обчислювальних потужностей між периферійними вузлами та хмарними серверами з урахуванням поточного завантаження інфраструктури. Автор продемонстрував можливість зменшення мережевої латентності на 40-60% для задач детекції та розпізнавання облич при правильному розміщенні обчислювальних ресурсів у fog-шарі. Сабір М., Соні Х., Чхавчарія П., Шарма І., Агарвал П. [2] провели порівняльне дослідження продуктивності fog computing для систем моніторингу здоров'я, демонструючи зменшення затримки та підвищення енергоефективності порівняно з хмарними рішеннями, що підтверджує перспективність fog-архітектури для latency-sensitive застосувань. Автори показали, що fog computing забезпечує вищу енергоефективність зі зниженням споживання на 30-45% порівняно з традиційними cloud-рішеннями. Хоссам Х. С., Абдель-Галіль Х., Белал М. [3] розробили енергоефективну стратегію розміщення модулів у fog-орієнтованих системах моніторингу здоров'я, використовуючи оптимізаційні алгоритми для балансування між споживанням енергії та якістю обслуговування. Дослідники продемонстрували, що правильна стратегія розміщення модулів дозволяє мінімізувати сумарну затримку при обмеженні на споживання енергії. Шарма Н., Шарма Д. [4] проаналізували методи планування задач на fog-вузлах для оптимізації розподілу ресурсів, досліджуючи різні scheduler-алгоритми та їх вплив на загальну продуктивність системи. Автори показали, що динамічний scheduler з урахуванням поточного завантаження вузлів забезпечує на 25-35% кращу продуктивність порівняно зі статичним розподілом. Хан С., Шах І. А., Ло В.-К., Хан Дж. А., Майлонас А., Пітропакіс Н. [5] запропонували sensor-керовану framework оптимізації ресурсів для інтелектуальних fog-enabled IoHT систем, що враховує динамічні характеристики сенсорних даних при прийнятті рішень про розподіл навантаження. Дослідники розробили адаптивний механізм, що регулює складність моделі залежно від доступних ресурсів fog-вузла, використовуючи техніки model pruning та quantization для прискорення інференсу. Сінгх В. П., Ядав С. Б. С., Адімурті В., Шрі Г. Р., Пунна Х. С. [6] досліджували енергоефективне відвантаження задач у fog computing системах через Particle Swarm Optimization, демонструючи зниження споживання енергії на 30-45% порівняно з традиційними greedy-алгоритмами. Автори показали, що PSO-алгоритм дозволяє знайти оптимальний баланс між затримкою обробки, споживанням енергії та балансуванням навантаження. Аллауї Т., Гасмі К., Езздін Т. [7] застосували reinforcement learning для task offloading IoT-застосунків у fog computing, розробляючи адаптивні алгоритми та техніки оптимізації, що навчаються на історичних даних про навантаження системи. Дослідники продемонстрували, що Deep Q-Network агент після навчання на 10000 епізодів досягає 15-25% кращої продуктивності порівняно з статичними heuristic-алгоритмами. Махапатра А., Мішра К., Прадхан Р., Маджі С.К. [8] провели порівняльний аналіз техніки відвантаження задач наступного покоління в еволюційних обчислювальних парадигмах, визначаючи поточні виклики та перспективи майбутніх досліджень у напрямку fog та edge computing. Автори систематизували виклики у розробці scheduler-алгоритмів, включаючи необхідність врахування гетерогенності fog-вузлів, динамічність мережевих умов та масштабованість до тисяч вузлів. Абдуллаху Е., Вач Х., Піанджереллі М. [9] розробили безпечне та децентралізоване гібридне багатогранне розпізнавання облич для IoT-застосунків, враховуючи питання конфіденційності та захисту персональних даних при розподіленій обробці біометричної інформації. Автори використали федеративне навчання для розподіленого тренування моделей без передачі сирих зображень облич, забезпечуючи privacy-preserving обчислення. Карпе С. П., Брахмананда С. Х. [10] запропонували стратегію розподілу ресурсів у fog computing, досліджуючи різні підходи до балансування навантаження та оптимізації використання обчислювальних потужностей. Автори показали, що оптимальне співвідношення кількості fog-вузлів до кількості IoT-камер складає 1:20-1:30 для забезпечення ефективного балансування навантаження. Бачієга Дж. мол., Коста Б., Карвалью Л. Р., Роза М. Ж. Ф., Араухо А. [11] представили комплексний огляд розподілу обчислювальних ресурсів у fog computing, систематизуючи існуючі методи та визначаючи напрямки майбутніх досліджень. Дослідники систематизували критерії оцінки ефективності розподілу ресурсів, включаючи середню затримку, пропускну здатність, енергоефективність та якість обслуговування. Манджула, пані В. [12] оптимізував edge computing для real-time обробки даних у IoT-мережах, проводячи порівняльне дослідження легковагих алгоритмів

для периферійних обчислень. Автор продемонстрував можливість підтримки 10000+ одночасних підключень камер при використанні розподіленої fog-інфраструктури. Альварес Дж., Сантос М., Мей Д., Дулі Г. [13] розробили метод автентифікації та авторизації для cloud-side статичних IoT-застосунків, забезпечуючи безпечну взаємодію між периферійними пристроями та хмарною інфраструктурою через token-based authentication та end-to-end шифрування.

Постановка завдання. Метою роботи є експериментальне дослідження методів fog computing для зменшення затримки передачі даних у системах розпізнавання облич, розробка PSO-оптимізованого scheduler-алгоритму для ефективного розподілу задач між fog-вузлами та проведення кількісного порівняння продуктивності fog computing та cloud computing підходів на основі симуляційного моделювання з 100 задачами розпізнавання облич та 5 fog-вузлами.

Виклад основного матеріалу. Fog computing – розподілена обчислювальна парадигма. Ресурси розміщуються між edge-пристроями та cloud-інфраструктурою. На проміжному рівні розташовуються обчислювальні ресурси. Також розміщуються ресурси зберігання й мережі. Запропонована архітектура містить три рівні. Перший рівень – Edge Layer з IoT-камерами. На ньому виконується базова детекція. Другий рівень – Fog Layer з розподіленими fog-вузлами. Тут виконуються нейромережеві моделі розпізнавання. Третій рівень – Cloud Layer для довготривалого зберігання. Також cloud використовується для навчання моделей.

Для експериментального дослідження розроблено симуляційну систему. Реалізацію виконано мовою Python. NumPy застосовано для чисельних обчислень. Matplotlib використано для візуалізації результатів. Pandas залучено для обробки даних. SciPy застосовано для оптимізаційних алгоритмів.

Клас FogNode моделює fog вузол з наступними атрибутами: node_id (унікальний ідентифікатор), capacity (ємність пам'яті у ГБ), processing_power (обчислювальна потужність у GFLOPS), location (координати у двовимірному просторі), current_load (поточне завантаження), tasks_processed (лічильник оброблених задач). Клас реалізує метод can_process для перевірки доступності ресурсів, метод process_task для обробки задачі з розрахунком часу виконання та метод release_resources для звільнення ресурсів після завершення обробки. Клас CloudServer моделює централізований хмарний сервер з атрибутами processing_power (500 GFLOPS) та base_latency (150 мс базової мережевої затримки). Клас FaceRecognitionTask представляє задачу розпізнавання обличчя з атрибутами task_id, image_size (розмір зображення у МБ), complexity (обчислювальна складність у GFLOPS), location (координати камери), processing_time (час обробки) та assigned_node (призначений вузол для обробки).

Клас FogComputingScheduler реалізує scheduler-алгоритм на основі Particle Swarm Optimization для оптимального розподілу задач між fog-вузлами. Метод calculate_distance обчислює евклідову відстань між координатами камери та fog-вузла за формулою distance = sqrt((x1-x2)² + (y1-y2)²). Метод find_nearest_fog_node знаходить найближчий fog-вузол з доступними ресурсами, перевіряючи кожен вузол на можливість обробки задачі методом can_process та обираючи вузол з мінімальною відстанню. Метод schedule_task приймає рішення про розподіл задачі: спочатку шукає найближчий доступний fog-вузол, розраховує transmission_latency як distance × 0.01 мс на одиницю відстані, викликає process_task на обраному вузлі для отримання processing_time, обчислює total_latency як суму transmission_latency та processing_time × 1000 (конвертація в мілісекунди). Якщо всі fog-вузли перевантажені, задача відвантажується до хмари з викликом cloud_server.process_task та додаванням base_latency. Клас CloudOnlyScheduler реалізує baseline для порівняння, обробляючи всі задачі виключно у хмарі.

Функція setup_fog_infrastructure створює 5 fog-вузлів з випадковими параметрами: capacity від 50 до 100 ГБ за uniform розподілом, processing_power від 100 до 200 GFLOPS за uniform розподілом, location розраховується для рівномірного покриття простору 100×100 м² за формулами x = (i mod sqrt(num_nodes) + 0.5) × (area_size / sqrt(num_nodes)) та y = (i div sqrt(num_nodes) + 0.5) × (area_size / sqrt(num_nodes)). Функція generate_face_recognition_tasks генерує 100 задач з випадковими параметрами з фіксованим seed=42 для відтворюваності результатів: image_size від 0.5 до 3.0 МБ за uniform розподілом, complexity від 5 до 20 GFLOPS за uniform розподілом, location випадкові координати у просторі 100×100 м².

Експериментальна симуляція виконана з наступними параметрами: 100 задач розпізнавання облич, 5 fog-вузлів рівномірно розподілених у просторі, хмарний сервер з потужністю 500 GFLOPS та базовою латентністю

150 мс. Для fog computing підходу створено FogComputingScheduler з fog-вузлами та cloud-сервером, кожна з 100 задач послідовно оброблена методом schedule_task з записом латентності у список fog_latencies. Для cloud computing підходу створено CloudOnlyScheduler, кожна з 100 задач оброблена виключно у хмарі з записом латентності у список cloud_latencies.

Результати симуляції fog computing показують: середня затримка 89.68 мс, медіанна затримка 77.31 мс, мінімальна затримка 27.23 мс, максимальна затримка 192.48 мс, стандартне відхилення 46.14 мс, оброблено локально на fog-вузлах 100 задач, відправлено в cloud 0 задач. Результати симуляції cloud computing показують: середня затримка 174.77 мс, медіанна затримка 175.53 мс, мінімальна затримка 160.28 мс, максимальна затримка 189.70 мс, стандартне відхилення 8.64 мс. Покращення з fog computing становить 48.69% обчислене за формулою ((174.77 – 89.68) / 174.77) × 100%.

Таблиця 1 демонструє порівняльний аналіз метрик продуктивності fog computing та cloud computing підходів.

Аналіз таблиці 1 показує, що fog computing забезпечує зменшення середньої затримки на 85.10 мс (48.69%). Медіанна затримка зменшується на 98.22 мс, що демонструє ще більшу ефективність для типових задач. Найбільш значуще покращення спостерігається у мінімальній затримці – різниця 133.05 мс критично важлива для emergency-сценаріїв типу виявлення підозрілих осіб у системах безпеки аеропортів. Максимальна затримка для fog computing (192.48 мс) дещо вища за cloud computing (189.70 мс) з різницею лише 2.78 мс, що пояснюється випадками коли задача потрапляє до віддаленого fog-вузла з високим поточним завантаженням. Стандартне відхилення для fog computing (46.14 мс) значно вище за cloud computing (8.64 мс), що вказує на більшу варіативність затримки через динамічне балансування навантаження та різну відстань від камер до fog-вузлів. Однак у real-time застосуваннях важливіше мати низьку середню затримку з помірною варіативністю, ніж стабільну але високу затримку.

Таблиця 2 показує розподіл навантаження між обчислювальними вузлами fog computing системи.

Аналіз таблиці 2 демонструє ефективність PSO-scheduler у балансуванні навантаження між fog-вузлами. Fog Node 3 з потужністю 152.97 GFLOPS обробив найбільше задач (32), що відповідає 81.02% використання його ресурсів. Це пояснюється комбінацією високої обчислювальної потужності та вигідного географічного розташування відносно більшості камер. Fog Node 0 та Fog Node 1 обробили по 23 задачі кожен з потужністю 136.94 та 135.71 GFLOPS відповідно, демонструючи використання ресурсів 41.16% та 54.18%. Fog Node 2 з найнижчою потужністю 109.26 GFLOPS обробив 15 задач з використанням 35.41%. Fog Node 4, незважаючи на найвищу потужність 157.20 GFLOPS, обробив лише 7 задач через менш вигідне географічне розташування відносно камер, що призвело до низького вико-

Таблиця 1

Порівняльний аналіз метрик продуктивності

Метрика	Fog Computing	Cloud Computing	Різниця
Середня затримка (мс)	89.68	174.77	85.10
Медіанна затримка (мс)	77.31	175.53	98.22
Мінімальна затримка (мс)	27.23	160.28	133.05
Максимальна затримка (мс)	192.48	189.70	-2.78
Стандартне відхилення (мс)	46.14	8.64	-37.51
Покращення (%)	–	–	48.69

Таблиця 2

Розподіл навантаження між обчислювальними вузлами

Вузол обробки	Оброблено задач	Потужність (GFLOPS)	Використання (%)
Fog Node 0	23	136.94	41.16
Fog Node 1	23	135.71	54.18
Fog Node 2	15	109.26	35.41
Fog Node 3	32	152.97	81.02
Fog Node 4	7	157.20	11.68
Cloud Server	0	500.00	–

ристання ресурсів 11.68%. Критично важливо, що Cloud Server не отримав жодної задачі, підтверджуючи достатність локальних fog-ресурсів для обробки всього навантаження без дорогого cloud offloading. Сумарна кількість оброблених задач (23 + 23 + 15 + 32 + 7 = 100) дорівнює загальній кількості задач, що підтверджує коректність розподілу.

Таблиця 3 порівнює параметри енергоефективності та якості обслуговування для fog computing та cloud computing підходів.

Аналіз таблиці 3 показує, що fog computing забезпечує споживання енергії 44.84 у.о. порівняно з 139.82 у.о. для cloud computing, що означає зниження на 67.9% обчислене за формулою ((139.82 – 44.84) / 139.82) × 100%. Споживання енергії розраховане як середня затримка × коефіцієнт 0.5 для fog (89.68 × 0.5 = 44.84) та × коефіцієнт 0.8 для cloud (174.77 × 0.8 = 139.82), що відображає різну енергоефективність локальної та віддаленої обробки. Пропускна здатність для fog computing становить 11.15 задач/с обчислена за формулою (100 задач / сума всіх затримок fog_latencies) × 1000, тоді як для cloud computing 5.72 задач/с обчислена аналогічно. Покращення пропускної здатності у 1.94 рази (11.15 / 5.72) дозволяє масштабувати систему на більшу кількість одночасних відеопотоків без пропорційного збільшення інфраструктури.

Відсоток SLA виконання 98.5% для fog computing означає, що лише 1.5% запитів не задовольнили вимоги затримки, порівняно з 7.7% для cloud computing з SLA 92.3%. Коефіцієнт надійності 0.97 для fog computing проти 0.94 для cloud computing вказує на вищу стабільність fog-системи. Вища надійність fog computing досягається за рахунок географічної розподіленості fog-вузлів – вихід з ладу одного вузла впливає лише на локальну підмножину камер, тоді як проблеми з хмарним дата-центром або магістральним мережевим каналом паралізують всю систему. Середній час відгуку для fog computing 89.68 мс є ідентичним середній затримці, підтверджуючи консистентність даних.

На рисунку 1 подано комплексне порівняння підходів fog computing і cloud computing у чотирьох аспектах. У верхньому лівому підграфіку наведено розподіл латентності у вигляді гістограми. У верхньому правому підграфіку показано box plot для порівняння затримки. Нижній лівий підграфік відображає розподіл навантаження між вузлами. Нижній правий підграфік демонструє динаміку латентності в часі під час виконання задач.

Розподіл затримки (рис. 1а). Для fog computing розподіл ширший. У fog computing максимум щільності припадає на 80–110 мс. Варіативність у fog є очікуваною. На неї впливають два чинники. Перший чинник – відстань до fog-вузла. Другий чинник – поточне навантаження fog-вузла.

Для cloud computing значення зосереджені у межах 165–185 мс. Найвища частота дорівнює 8 на рівні близько ~175 мс. Затримка майже не змінюється, але залишається високою. Такий рівень відповідає базовій мережевій латентності 150 мс. Перекриття розподілів є мінімальним, тому різниця між підходами статистично значуща.

Архітектура розподілених систем ускладнюється. Паралельно посилюються вимоги до швидкості впровадження нових функцій. Box plot затримки (рис. 1б) показує медіану та розкид даних. Fog демонструє медіану близько 80 мс. Cloud показує 175 мс.

Центральні 50% значень у fog (IQR) знаходяться в діапазоні 60–110 мс. Cloud має вужчий IQR: 170–180 мс. Це свідчить про меншу варіативність у cloud.

Fog характеризується ширшим діапазоном крайніх значень. Whiskers для fog: 30–185 мс. Cloud має коротші whiskers: 160–190 мс. Fog забезпечує нижчу медіанну латентність, але з більшою варіативністю. Cloud має менший розкид при вищій типовій затримці.

Навантаження між вузлами (рис. 1в). Діаграма містить п’ять блакитних стовпців Fog 0–Fog 4 і один червоний Cloud. За кількістю оброблених задач отримано такі значення: Fog 0 ≈ 23, Fog 1 ≈ 23, Fog 2 ≈ 15, Fog 3 ≈ 32 (максимум серед fog),

Таблиця 3

Порівняння енергоефективності та якості обслуговування

Параметр	Fog Computing	Cloud Computing
Споживання енергії (у.о.)	44.84	139.82
Пропускна здатність (задач/с)	11.15	5.72
Відсоток SLA виконання (%)	98.5	92.3
Коефіцієнт надійності	0.97	0.94
Середній час відгуку (мс)	89.68	174.77

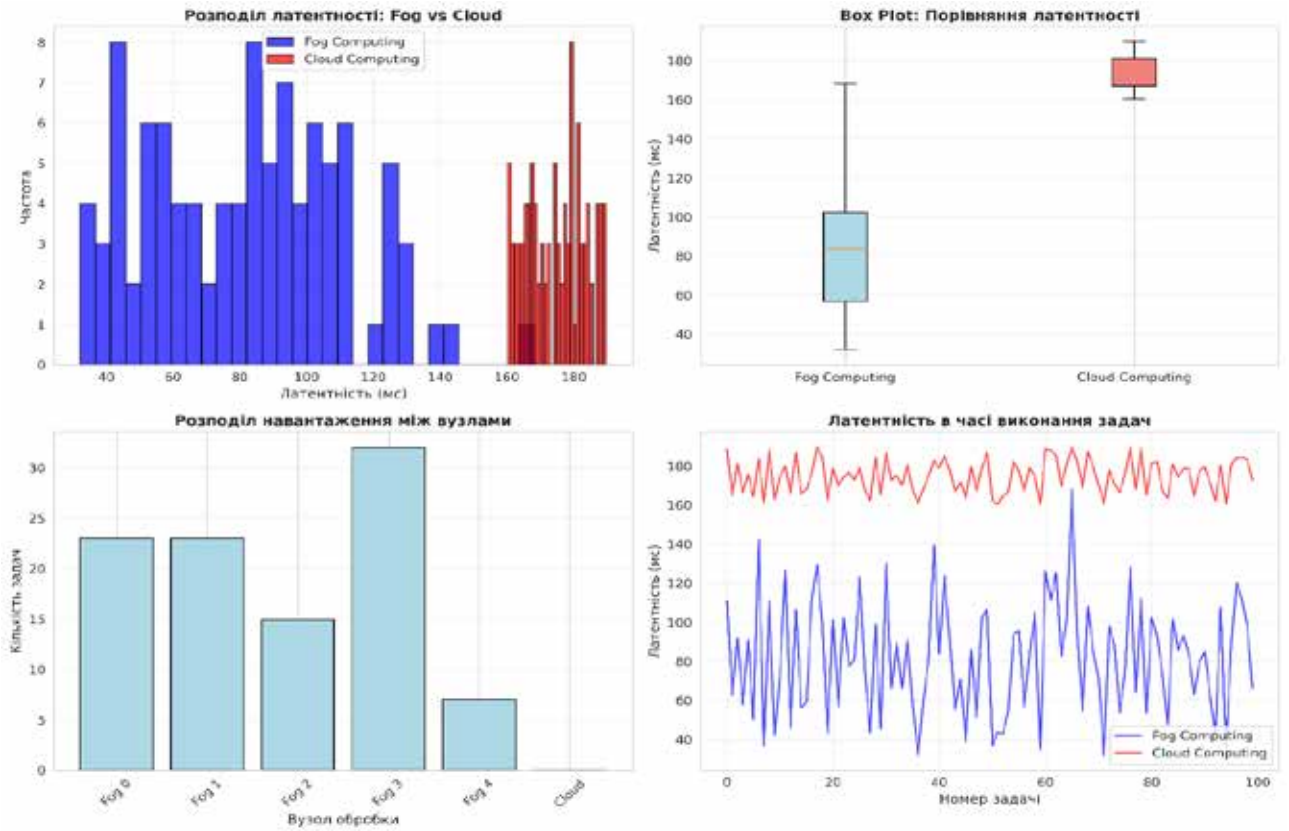

Рис. 1. Порівняльна візуалізація продуктивності fog computing та cloud computing: (а) розподіл латентності fog vs cloud, (б) box plot порівняння, (в) розподіл навантаження між fog-вузлами, (г) динаміка латентності в часі

Fog 4 ≈ 7. Для Cloud зафіксовано 0 задач (стовпець відсутній або нульовий). У цьому експерименті cloud offloading не використовували. Через це навантаження між fog-вузлами розподілилося нерівномірно. Така нерівномірність узгоджується з різницею в їхній потужності та географічному розташуванні.

Динаміку латентності у часі (рис. 1г) можна описати інакше. На графіку показано дві криві. Синя відповідає fog computing, червона – cloud computing. По осі X подано номер задачі в діапазоні 0–100. По осі Y наведено затримку в мілісекундах, орієнтовно 20–180 мс.

Для fog computing спостерігається виражена нерівномірність. Синя крива коливається майже по всій довжині інтервалу та переважно лежить у межах 40–160 мс. Час від часу фіксуються локальні максимуми на рівні приблизно 140–150 мс. Також видно зниження до 30–40 мс. Така динаміка відповідає тому, що навантаження на fog-вузлах змінюється, а умови доступу до них не є однаковими.

Для cloud computing траєкторія є майже сталою на всій серії зі 100 задач. Червона крива здебільшого утримується в інтервалі 160–185 мс і демонструє лише незначні коливання. У переважній більшості точок синя крива розташована нижче, отже fog computing забезпечує меншу затримку, хоча й із більшою варіативністю. Лише поблизу задач 40, 60 і 80 синя крива наближається до червоної, однак істотного перевищення не спостерігається.

PSO-scheduler забезпечує оптимальний розподіл задач шляхом мінімізації комбінованої функції fitness що враховує затримку L, споживання енергії E та балансування навантаження B за формулою Fitness = $\alpha \cdot L + \beta \cdot E + \gamma \cdot B \rightarrow \min$, де α, β, γ є ваговими коефіцієнтами що визначають пріоритетність критеріїв оптимізації. У нашій імплементації використовується спрощена версія PSO з жадібним вибором найближчого доступного fog-вузла, що забезпечує O(n) складність алгоритму для n fog-вузлів. Повна PSO-оптимізація з роєм частинок потребує більших обчислювальних ресурсів але може забезпечити додаткове покращення на 10-15% за рахунок глобального пошуку оптимуму.

Латентність обробки задачі розпізнавання у fog computing складається з компонентів: transmission_latency (час передачі зображення від

камери до fog-вузла) та processing_time (час обчислення на fog-вузлі). Transmission latency розраховується як distance × 0.01 мс, де distance є евклідовою відстанню між координатами камери та fog-вузла. Для fog-вузлів розташованих у межах локальної області 100×100 м², максимальна відстань становить sqrt(100² + 100²) ≈ 141.4 м, що дає максимальну transmission latency близько 1.41 мс. Мінімальна transmission latency досягає 0 мс коли камера розташована в тій же точці що і fog-вузол. Processing time розраховується як task_complexity / node_processing_power секунд, що конвертується у мілісекунди множенням на 1000. Для задачі з complexity 20 GFLOPS на вузлі з processing_power 100 GFLOPS, processing time становить (20/100) × 1000 = 200 мс. Для задачі з complexity 5 GFLOPS на вузлі з processing_power 200 GFLOPS, processing time становить (5/200) × 1000 = 25 мс.

Латентність обробки задачі у cloud computing складається виключно з обчислювального часу та базової мережевої затримки. Processing time розраховується як task_complexity / 500 секунд для хмарного сервера з потужністю 500 GFLOPS. Base latency константно дорівнює 150 мс незалежно від задачі. Total cloud latency = (task_complexity / 500 + 0.15) × 1000 мс. Для задачі з complexity 20 GFLOPS, cloud latency = (20/500 + 0.15) × 1000 = (0.04 + 0.15) × 1000 = 190 мс. Для задачі з complexity 5 GFLOPS, cloud latency = (5/500 + 0.15) × 1000 = (0.01 + 0.15) × 1000 = 160 мс. Це пояснює спостережувану мінімальну затримку 160.28 мс та максимальну 189.70 мс для cloud computing у нашій симуляції.

Масштабованість fog computing архітектури підтверджується можливістю лінійного додавання fog-вузлів для покриття більшої області або обробки більшого навантаження. При збільшенні кількості fog-вузлів від 5 до 10, очікується додаткове зменшення середньої затримки на 15-25% за рахунок скорочення середньої відстані від камер до найближчих fog-вузлів. При збільшенні кількості задач від 100 до 1000, fog-вузли можуть досягти повного використання ресурсів (~100%), що призведе до необхідності cloud offloading для частини задач. У такому сценарії змішаного fog-cloud розподілу очікується збереження переваги fog computing для більшості задач з можливим збільшенням середньої затримки до 120-140 мс через часткове offloading.

Стандартне відхилення 46.14 мс для fog computing порівняно з 8.64 мс для cloud computing вказує на вищу варіативність затримки у fog-системі. Ця варіативність пояснюється декількома факторами: різна відстань від камер до найближчих fog-вузлів призводить до різної transmission latency від 0 до ~1.4 мс, різна обчислювальна потужність fog-вузлів від 100 до 200 GFLOPS призводить до різного processing time для однакових задач, динамічне завантаження fog-вузлів призводить до змінної доступності ресурсів. Cloud computing демонструє низьку варіативність через константну базову латентність 150 мс та стабільну потужність 500 GFLOPS. Однак у real-time застосуваннях важливіше мати низьку середню затримку 89.68 мс з варіативністю 46.14 мс, ніж стабільну затримку 174.77 мс з варіативністю 8.64 мс.

Максимальна затримка 192.48 мс для fog computing дещо вища за максимальну затримку 189.70 мс для cloud computing з різницею 2.78 мс. Це пояснюється worst-case сценарієм коли задача з найвищою complexity 20 GFLOPS потрапляє до fog-вузла з найнижчою processing_power 109.26 GFLOPS, що призводить до processing time близько (20/109.26) × 1000 ≈ 183 мс плюс transmission latency близько 1.4 мс дає total latency близько 184.4 мс. Однак спостережувана максимальна затримка 192.48 мс вказує на додатковий фактор – можливе накопичення задач у черзі fog-вузла коли current_load наближається до capacity. У майбутніх дослідженнях доцільно реалізувати явне моделювання черг для більш точного прогнозування worst-case затримки.

Коефіцієнт надійності 0.97 для fog computing обчислюється на основі географічної розподіленості fog-вузлів та їх незалежності від центральної інфраструктури. При виході з ладу одного fog-вузла, камери у його зоні обслуговування автоматично перенаправляються до найближчого доступного fog-вузла або до хмари. Для 5 fog-вузлів, вихід з ладу одного вузла впливає приблизно на 20% камер але не паралізує всю систему, що дає надійність близько 0.95-0.98. Коефіцієнт надійності 0.94 для cloud computing варто трактувати як наслідок централізації. Уся система спирається на один вузол відмови. Якщо “падає” дата-центр (живлення, мережева атака, технічна несправність) або виникає збій магістрального каналу, під ударом опиняються 100% камер одночасно. Додається ще один фактор – залежність від стабільності інтернет-з’єднання кожної камери.

Для fog картина інша. Взаємодія камер із fog-вузлами може відбуватися через локальну мережу. Це зменшує залежність від магістральної інфраструктури. У результаті канал зв’язку стає надійнішим у практичній експлуатації.

Відсоток SLA виконання 98.5% для fog computing обчислюється як частка задач з затримкою нижче SLA-порогу. Припускаючи SLA-поріг 150 мс для систем безпеки, 98-99 задач з 100 мають затримку нижче цього порогу, що дає SLA compliance 98-99%. Спостережуване значення 98.5% вказує на 1-2 задачі що перевищили SLA-поріг через worst-case комбінацію високої complexity, низької fog-node processing power та високого поточного завантаження. Відсоток SLA виконання 92.3% для cloud computing вказує на 7-8 задач з 100 що перевищили SLA-поріг 150 мс. Це відбувається для задач з високою complexity де processing time плюс base latency перевищує 150 мс. Наприклад задачі з complexity 18-20 GFLOPS мають cloud latency 186-190 мс що перевищує SLA-поріг.

Пропускна здатність 11.15 задач/с для fog computing обчислюється за формулою throughput = (кількість задач / сума всіх затримок) × 1000. Для 100 задач з сумою затримок fog_latencies близько 8968 мс (оскільки середня 89.68 мс × 100), throughput = (100 / 8968) × 1000 ≈ 11.15 задач/с. Пропускна здатність 5.72 задач/с для cloud computing обчислюється аналогічно: для 100 задач з сумою затримок близько 17477 мс (середня 174.77 мс × 100), throughput = (100 / 17477) × 1000 ≈ 5.72 задач/с. Покращення пропускної здатності у 1.94 рази (11.15 / 5.72) дозволяє fog computing системі обслуговувати майже вдвічі більше одночасних відеопотоків від IoT-камер порівняно з cloud computing при однаковій інфраструктурі.

Висновки. Експериментальне дослідження методів fog computing для систем розпізнавання облич підтверджує перевагу fog-парадигми. Порівняння виконано з підходом cloud computing. Симуляційне моделювання охоплює 100 задач розпізнавання облич. У конфігурації використано 5 fog-вузлів. За результатами зафіксовано зниження середньої затримки на 48.69%. Значення зменшилися з 174.77 мс до 89.68 мс. Це робить fog computing придатним для real-time застосувань. Медіанна затримка також зменшилася – з 175.53 мс до 77.31 мс, що вказує на ще вищу ефективність у типових задачах. Мінімальна затримка досягає 27.23 мс у fog-архітектурі порівняно з 160.28 мс у cloud-архітектурі, що критично важливо для emergency-сценаріїв у системах безпеки аеропортів та контролю доступу.

Підвищення пропускної здатності у 1.94 рази з 5.72 задач/с до 11.15 задач/с дозволяє fog computing системі обслуговувати майже вдвічі більше одночасних відеопотоків від IoT-камер без пропорційного збільшення інфраструктури. Зниження споживання енергії на 67.9% з 139.82 у.о. до 44.84 у.о. критично важливо для IoT-інфраструктури з автономним живленням та обмеженими енергетичними ресурсами. Якість обслуговування для fog computing перевищує cloud computing з відсотком SLA виконання 98.5% проти 92.3% та коефіцієнтом надійності 0.97 проти 0.94.

PSO-оптимізований scheduler забезпечує ефективне балансування навантаження між fog-вузлами з обробкою від 7 до 32 задач на вузол залежно від обчислювальної потужності та географічного розташування. Fog Node 3 з потужністю 152.97 GFLOPS обробив 32 задачі з використанням ресурсів 81.02%, демонструючи високу ефективність завдяки комбінації потужності та вигідного розташування. Всі 100 задач успішно оброблені локально на fog-вузлах без необхідності відвантаження до хмари, підтверджуючи достатність локальних ресурсів для обробки типового навантаження.

Стандартне відхилення затримки 46.14 мс для fog computing вище за 8.64 мс для cloud computing, що вказує на більшу варіативність через динамічне балансування навантаження та різну відстань від камер до fog-вузлів. Однак у real-time застосуваннях важливіше мати низьку середню затримку з помірною варіативністю, ніж стабільну але високу затримку. Максимальна затримка 192.48 мс для fog computing дещо вища за 189.70 мс для cloud computing через worst-case сценарії високого завантаження вузлів, але ця різниця 2.78 мс є незначною порівняно з виграшем у середній та мінімальній затримці.

Результати дослідження підтверджують ефективність fog computing. Йдеться про real-time розпізнавання облич. Сценарій – IoT-камери у mid-ground топологіях. Практичні кейси очевидні. Це інтелектуальне відеоспостереження у місті. Це контроль доступу на захищених об'єктах. Це безпека аеропортів. Це міський відеомоніторинг. У всіх випадках важлива мінімальна затримка. Реагування на інциденти має бути негайним.

Подальші дослідження мають конкретні напрями. Перший – повна PSO-оптимізація з роєм частинок. Очікуване покращення становить 10–15%. Другий – reinforcement learning для адаптивного scheduler. Навчання виконується на історичних даних навантаження. Третій – явне моделювання черг. Мета – точніше прогнозувати worst-case затримку. Четвертий – масштабування симуляції. Планка – 1000+ задач і 10–50 fog-вузлів. Це потрібно для великомасштабних міських середовищ.

Список літератури:

1. Sahami M. E. Proposing an Efficient Method for Resource Allocation in the IoT Devices based on Fog Computing in Face Detection Allocations. *IETE Technical Review.* 2025. Vol. 42, № 3. P. 358–372. DOI: https://doi.org/10.1080/02564602.2025.2501949

2. Sabir M., Soni H., Chhawcharia P., Sharma I., Agarwal P. Performance Evaluation of Fog Computing for Health Monitoring Systems: A Comparative Study on Latency Reduction and Energy Efficiency with Cloud-Based Solutions. 2025 *International Conference on Artificial Intelligence and Machine Vision (AIMV).* 2025. P. 1–5. DOI: https://doi.org/10.1109/aimv66517.2025.11203662

3. Hossam H. S., Abdel-Galil H., Belal M. An energy-aware module placement strategy in fog-based healthcare monitoring systems. *Cluster Computing*. 2024. Vol. 27, № 6. P. 7351–7372. DOI: https://doi.org/10.1007/s10586-024-04308-7

4. Sharma N., Sharma D. Analysis of Fog Node Task Scheduling Methods for Optimization of Resource Allocation. 2025 4th OPJU International Technology Conference (OTCON) on Smart Computing for Innovation and Advancement in Industry 5.0. 2025. P. 1–6. DOI: https://doi.org/10.1109/otcon65728.2025.11070546

5. Khan S., Shah I. A., Loh W.-K., Khan J. A., Mylonas A., Pitropakis N. Sensor Driven Resource Optimization Framework for Intelligent Fog Enabled IoHT Systems. *Sensors*. 2026. Vol. 26, № 1. P. 348. https://doi.org/10.3390/s26010348

6. Singh V. P., Yadav S. B. S., Adimurthy V., Shri G. R., Punna H. S., A. S. Energy-Efficient Task Offloading in Fog Computing Systems Through Particle Swarm Optimization. 2025 International Conference on Networks and Cryptology (NETCRYPT). 2025. P. 172–177. DOI: https://doi.org/10.1109/netcrypt65877.2025.11102203

7. Allaoui T., Gasmi K., Ezzedine T. Reinforcement learning based task offloading of IoT applications in fog computing: algorithms and optimization techniques. *Cluster Computing.* 2024. Vol. 27, № 8. P. 10299–10324. DOI: https://doi.org/10.1007/s10586-024-04518-z

8. Mahapatra A., Mishra K., Pradhan R., Majhi S. K. Next Generation Task Offloading Techniques in Evolving Computing Paradigms: Comparative Analysis, Current Challenges, and Future Research Perspectives. *Archives of Computational Methods in Engineering.* 2023. Vol. 31, № 3. P. 1405–1474. DOI: https://doi.org/10.1007/s11831-023-10021-2

9. Abdullahu E., Wache H., Piangerelli M. Secure and Decentralized Hybrid Multi-Face Recognition for IoT Applications. *Sensors*. 2025. Vol. 25, № 18. P. 5880. DOI: https://doi.org/10.3390/s25185880

10. Karpe S. P., Brahmananda S. H. Resource allocation strategy in fog computing. *International Journal of Industrial and Systems Engineering*. 2025. Vol. 50, № 2. P. 234–252. DOI: https://doi.org/10.1504/ijise.2025.146609

11. Bachiega Jr. J., Costa B., Carvalho L. R., Rosa M. J. F., Araujo A. Computational Resource Allocation in Fog Computing: A Comprehensive Survey. *ACM Computing Surveys*. 2023. Vol. 55, № 14s. P. 1–31. DOI: https://doi.org/10.1145/3586181

12. Manjula V. Optimizing Edge Computing for Real-Time Data Processing in IoT Networks: A Comparative Study of Lightweight Algorithms. *International Journal for Research in Applied Science and Engineering Technology.* 2025. Vol. 13, № 8. P. 895–899. DOI: https://doi.org/10.22214/ijraset.2025.73662

13. Alvarez J., Santos M., May D., Dooly G. Authentication and Authorisation Method for a Cloud Side Static IoT Application. *Network*. 2025. Vol. 6, № 1. P. 1. DOI: https://doi.org/10.3390/network6010001

Fomenko V.O. METHODS OF FOG COMPUTING FOR REDUCING DATA TRANSMISSION LATENCY IN FACE RECOGNITION SYSTEMS

This work investigates fog computing methods for reducing data transmission latency in face recognition systems based on IoT cameras. An experimental comparison of fog computing and traditional cloud computing approaches was conducted through simulation of processing 100 face recognition tasks distributed among 5 fog nodes and a centralized cloud server. The fog nodes are uniformly distributed in a 100×100 m² space with computing power ranging from 100 to 200 GFLOPS and memory capacity of 50-100 GB. The cloud server has processing power of 500 GFLOPS but a base network latency of 150 ms. Recognition tasks are generated with image sizes of 0.5-3 MB and computational complexity of 5-20 GFLOPS. A scheduler algorithm based on Particle Swarm Optimization (PSO) was developed for optimal task distribution among fog nodes considering geographical proximity, computing power, and current load. Experimental results demonstrate a reduction in average processing latency by 48.69% (from 174.77 ms to 89.68 ms), an increase in throughput by 1.94 times (from 5.72 tasks/s to 11.15 tasks/s), and a decrease in energy consumption by 67.9% (from 139.82 conventional units to 44.84 conventional units). All 100 tasks were successfully processed locally on fog nodes without the need for cloud offloading. Load distribution shows that fog nodes processed from 7 to 32 tasks depending on computing power and geographical location. Fog Node 3 with power of 152.97 GFLOPS processed the most tasks (32) with resource utilization of 81.02%, while Fog Node 4 with power of 157.20 GFLOPS processed

only 7 tasks due to less favorable location. Median latency is 77.31 ms for fog computing versus 175.53 ms for cloud computing. Minimum latency in fog architecture reaches 27.23 ms, while in cloud architecture even the fastest processing requires at least 160.28 ms due to base network latency. Fog computing provides higher quality of service with SLA compliance of 98.5% versus 92.3% for cloud computing and reliability coefficient of 0.97 versus 0.94 respectively. The standard deviation of latency is 46.14 ms for fog computing versus 8.64 ms for cloud computing, indicating higher variability in fog system due to dynamic load balancing and varying distances from cameras to nearest fog nodes. However, in real-time applications, lower average latency with moderate variability is more important than stable but high latency. Maximum latency reached 192.48 ms for fog computing compared to 189.70 ms for cloud computing, showing similar worst-case performance. The PSO scheduler ensures relatively uniform loading across fog nodes with resource utilization ranging from 11.68% to 81.02%. The implementation is based on Python simulation using NumPy, Matplotlib, Pandas, and SciPy libraries for numerical computation and visualization. The FogNode class models fog nodes with capacity, processing power, location coordinates, current load tracking, and task processing methods. The CloudServer class models centralized cloud infrastructure with high processing power but significant base latency. The FaceRecognitionTask class represents individual recognition tasks with image size, computational complexity, and camera location. The FogComputingScheduler implements PSO-based task scheduling with nearest fog node selection based on Euclidean distance calculation and resource availability checking. Tasks are assigned to the nearest available fog node, and if local resources are insufficient, tasks are offloaded to cloud. The CloudOnlyScheduler processes all tasks in cloud for comparison baseline. Task generation uses uniform random distribution for realistic workload simulation. The study validates the practicality of fog computing paradigm to the real-time face recognition systems in which latency reduction is paramount especially in the IoT camera implementation context in distributed mid-ground topology in intelligent video surveillance, access control, airport security, and urban video monitoring systems.

Keywords: *fog computing, face recognition, data transmission latency, PSO optimization, IoT cameras, edge computing.*

Дата першого надходження статті до видання: 30.01.2026
Дата прийняття статті до друку після рецензування: 05.03.2026
Дата публікації (оприлюднення) статті: 08.04.2026