Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка програмного забезпечення для порівняльного аналізу обчислювальних можливостей квантових комп’ютерів на основі вентилів з використанням мови програмування Python

Виконала: студентка IV курсу, групи СПз-41
спеціальності 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

Жмуд А. О.
(підпис) (прізвище та ініціали)

Керівник Пастух О.А.
(підпис) (прізвище та ініціали)

Нормоконтроль Стоянов Ю. М.
(підпис) (прізвище та ініціали)

Завідувач кафедри Петрик М. Р.
(підпис) (прізвище та ініціали)

Рецензент
(підпис) (прізвище та ініціали)

Тернопіль
2026

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет Комп'ютерно інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра Програмної інженерії
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Петрик М. Р.
(підпис) (прізвище та ініціали)
« » 2026 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавр
(назва освітнього ступеня)

за спеціальністю 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

студентці Жмуд Ангеліні Олегівні
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка програмного забезпечення для порівняльного аналізу обчислювальних можливостей квантових комп'ютерів на основі вентилів з використанням мови програмування Python

Керівник роботи д.т.н., проф. Пастух О.А.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «___» _____ 2026 року № _____

2. Термін подання студентом завершеної роботи _____

3. Вихідні дані до роботи Предметна область, технічне завдання, вимоги та специфікація, програмне рішення

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. Розділ 1. Аналіз предметної області. Розділ 2. Проєктування. Розділ 3. Тестування Розділ 4. Безпека життєдіяльності та основи охорони праці. Висновок. Список використаних джерел. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
Діаграми, знімки екрану з проробленою роботою, ілюстративні зображення, інформативні зображення для доповнення тексту.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці			

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Ознайомлення з завданням до кваліфікаційної роботи		
2	Підбір джерел по темі кваліфікаційної роботи		
3	Опрацювання джерел інформації по темі кваліфікаційної роботи		
4	Розробка архітектури програмного рішення		
5	Ознайомлення з новими технологіями		
6	Розробка на основі опрацьованої інформації		
7	Виконання звіту по проробленій роботі		
8	Виконання завдання до підрозділу «Безпека життєдіяльності»		
9	Виконання завдання до підрозділу «Основи охорони праці»		
10	Оформлення кваліфікаційної роботи		
11	Нормоконтроль		
12	Перевірка на плагіат		
13	Попередній захист кваліфікаційної роботи		
14	Захист кваліфікаційної роботи		

Студент _____ Жмуд А. О.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Пастух О.А.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка програмного забезпечення для порівняльного аналізу обчислювальних можливостей квантових комп’ютерів на основі вентилів з використанням мови програмування Python // Кваліфікаційна робота освітнього рівня «Бакалавр» // Жмуд Ангеліна Олегівна // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп’ютерно-інформаційних систем і програмної інженерії, кафедра програмної інженерії, група СПз-41 // Тернопіль, 2026 // Ст. – 97, рис. – 23, табл. – 14 , додат. – 1, бібліогр. – 89.

Ключові слова: квантові обчислення, бенчмаркінг квантових комп'ютерів, дискримінація квантових вимірювань, дискримінація вимірювань фон Неймана, відкритий код, програмування на Python.

Головною метою цієї кваліфікаційної роботи є закладення дослідницького фундаменту шляхом інтеграції квантових обчислень із класичними периферійно-хмарними середовищами для підвищення продуктивності в низці прикладних задач, які активно досліджуються науковцями. Квантові комп'ютери стають важливим інструментом в обчислювальній галузі, використовуючи принципи квантової механіки для вирішення специфічних задач, що наразі перебувають поза межами можливостей класичних комп'ютерів. Ця технологія має значні перспективи для розгортання у периферійно-хмарних інфраструктурах, де вона може забезпечити обробку даних із низькою затримкою та захищений зв'язок.

Перший розділ присвячено аналізу умов успішного розгортання гібридних квантово-класичних периферійних хмар та обґрунтуванню вимог до всеосяжної системи оцінювання для забезпечення їхньої відповідності критеріям продуктивності. Запропоновано новий фреймворк для квантового бенчмаркінгу, що включає два окремі методи оцінювання показників затримки на основі рівнів квантової транспіляції на різних квантово-периферійно-хмарних платформах.

У другому розділі спершу описано результати експериментальної валідації запропонованого фреймворку для периферійно-хмарного середовища, яку було проведено шляхом тестування кількох відомих і корисних квантових алгоритмів,

потенційно ефективних у цій області, включаючи алгоритм Шора, алгоритм Гровера та алгоритм квантових блукань. За результатами тестування визначено оптимальний рівень транспіляції для досягнення максимальної продуктивності в квантово-периферійно-хмарних середовищах. Наведено критично важливі висновки щодо поточних та перспективних можливостей інтеграції квантових процесорів, які забезпечують всебічну оцінку їхнього потенціалу для підвищення ефективності периферійних хмар за мінливих параметрів, зокрема точності та рівнів транспіляції.

У третьому розділі проведено аналіз ергономічні проблеми безпеки життєдіяльності розробника програмного забезпечення з точки зору охорони праці та безпеки життєдіяльності, розглянуто значення автоматизації виробничих процесів в питаннях охорони праці. згідно з нормативними документами з охорони праці та безпеки життєдіяльності. Розглянуто долікарську допомогу при ураженні електричним струмом та інженерно-технічні рішення з охорони праці.

ABSTRACT

Developing software for comparative analysis of the computational capabilities of gate-based quantum computers using the Python programming language // Qualification Thesis for the Bachelor's Degree // Zhmud Anhelina Olehivna // Ternopil Ivan Puluj National Technical University, Faculty of Computer and Information Systems and Software Engineering, Department of Software Engineering, Group SPz-41 // Ternopil, 2026 // Pages – 97, Figures – 20, Tables – 8, Appendices – 1, References – 26.

Keywords: quantum computing, benchmarking quantum computers, discrimination of quantum measurements, discrimination of von Neumann measurements, open-source, Python programming.

The main goal of this qualification work is to lay a research foundation by integrating quantum computing with classical edge-cloud environments to improve performance in a number of applied problems that are actively studied by scientists. Quantum computers are becoming an important tool in the computing industry, using the principles of quantum mechanics to solve specific problems that are currently beyond the capabilities of classical computers. This technology has significant prospects for deployment in edge-cloud infrastructures, where it can provide low-latency data processing and secure communication.

The first section is devoted to the analysis of the conditions for the successful deployment of hybrid quantum-classical edge clouds and the justification of the requirements for a comprehensive evaluation system to ensure their compliance with performance criteria. A new framework for quantum benchmarking is proposed, which includes two separate methods for estimating latency metrics based on quantum transpilation levels on different quantum edge-cloud platforms.

The second section first describes the results of experimental validation of the proposed framework for edge-cloud environments, which was conducted by testing several well-known and useful quantum algorithms that are potentially effective in this area, including Shor’s algorithm, Grover’s algorithm, and quantum walk algorithm. Based on the testing results, the optimal transpilation level for achieving maximum

performance in quantum edge-cloud environments is determined. Critical conclusions are presented on current and future integration capabilities of quantum processors, which provide a comprehensive assessment of their potential for improving edge cloud performance under varying parameters, including accuracy and transpilation levels.

The third section analyzes the ergonomic problems of the software developer's life safety from the point of view of labor protection and life safety, considers the importance of automation of production processes in matters of labor protection. according to regulatory documents on labor protection and life safety. Considers pre-medical care for electric shock and engineering and technical solutions for labor protection.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

QPU — Quantum Processing Unit, квантовий процесор

NISQ — Noisy Intermediate-Scale Quantum, шумове квантове устаткування проміжного масштаб

CPUs/GPUs — Central Processing Units / Graphics Processing Units, центральні та графічні процесори

IoT — Internet of Things, інтернет речей

QS — Quantum Switches, квантові перемикачі

QR — Quantum Repeaters, квантові повторювачі

API — Application Programming Interface, інтерфейс прикладного програмування

SDK — Software Development Kit, набір засобів розробки програмного забезпечення

DFE — Direct Fidelity Estimation, пряме оцінювання точності

QPT — Quantum Set Tomography, томографія квантових наборів

MQT — Munich Quantum Toolkit, мюнхенський квантовий інструментарій

QUARK — Quantum Computing Application Benchmark, фреймворк тестування квантових додатків

ВСТУП

У наш час інтеграція квантових обчислювальних пристроїв (QPU) у класичні периферійно-хмарні інфраструктури (hybrid quantum-classical edge-cloud) є фундаментальною задачею для розвитку обчислювальних систем нового покоління, що працюють із великими масивами даних та складними високорозмірними завданнями оптимізації, криптографії і машинного навчання. Проте більшість сучасних методів аналізу продуктивності обмежуються ізольованим тестуванням суто апаратних показників (таких як кількість кубітів чи час когерентності) або запусками спрощених синтетичних тестів, ігноруючи складну розподілену структуру реальних мережевих взаємозв'язків, затримок трансляції схем та специфіку виконання алгоритмів у реальних розподілених середовищах. Серйозним викликом у розробці таких гетерогенних систем є точне архітектурне та програмне моделювання наскрізної продуктивності (end-to-end performance), де використання спрощених підходів не дозволяє в повній мірі відобразити складну природу взаємодії між класичними периферійними вузлами та обчислювальними пристроями ери NISQ (Noisy Intermediate-Scale Quantum).

Серед програмно-інженерних підходів до моделювання складних взаємозв’язків між компонентами розподілених обчислювальних середовищ значну увагу привертає концепція комплексного багаторівневого бенчмаркінгу з використанням канонічних ресурсомістких квантових алгоритмів (Шора, Гровера, квантових блукань) у поєднанні з операційними методами перевірки точності виконання операцій. Цей підхід дозволяє розглядати оцінку продуктивності як наскрізний процес профілювання затримок та аналізу здатності систем до розрізнення квантових станів (зокрема, через класичні та користувацькі вимірювання фон Неймана). Використання інтегральних оцінок латентності (latency scores) та рівнів компіляторної транспиляції (transpilation levels) дає можливість визначати моделі ефективності як логічні комбінації гетерогенних джерел метрик — статистичних даних виконання, мережевих затримок запитів та

показників точності (fidelity), що значно підвищує надійність проектування та розгортання прикладного програмного забезпечення.

Однак застосування таких комплексних бенчмаркінгових підходів в інженерії програмного забезпечення стикається з критичною проблемою масштабованості, високої вартості та обмеженої доступності фізичного квантового обладнання. Використання натурного тестування на реальних QPU та симуляторах різних хмарних провайдерів (таких як IBM Quantum Lab чи Amazon Braket) для побудови точних моделей продуктивності веде до експоненціального зростання обсягу необхідних тестових запусків, часу очікування у чергах та фінансових витрат через потребу перебору багатьох рівнів оптимізації компіляторів та врахування апаратних шумів. Це робить традиційні підходи прямого перебору конфігурацій обчислювально неефективними для реальних прикладних задач великої розмірності. Це створює серйозне вузьке місце при створенні прикладних програмних засобів та систем оркестрації для автоматизованого тестування гібридних обчислень.

Найбільш підходящою базою для вирішення цієї проблеми є розробка спеціалізованих кросплатформних алгоритмів та модульних програмних рішень, що дозволяють уникати повного перебору всіх можливих комбінацій параметрів компіляції та мережевого середовища. Для подолання проблеми експоненціальної ресурсної складності необхідно впроваджувати методи профілювання, що базуються на низькорівневих оцінках здатності пристроїв до розрізнення вимірювань під час їх побудови (наприклад, через операційні інтерфейси командного рядка) та автоматизованому аналізі зважених часових показників. У зв'язку з цим виникає гостра потреба у програмній реалізації загального каркаса (фреймворка), який би інтегрував різні методи аналізу продуктивності — на основі виконання канонічних алгоритмів, низькорівневого тестування квантових вимірювань та алгоритмів постпроцесорного коригування помилок зчитування (readout error mitigation) — у єдину систему автоматизованого бенчмаркінгу та оптимізації гібридних обчислень.

Актуальність теми роботи полягає у необхідності створення ефективної програмної платформи для комплексного бенчмаркінгу та оптимізації гібридних обчислень через процеси профілювання затримок та точності операцій, що дозволить автоматизувати вибір конфігурацій розгортання програмного забезпечення в периферійно-хмарних інфраструктурах, одночасно вирішуючи проблему високої обчислювальної та ресурсної складності тестування у квантових просторах ери NISQ.

Метою роботи є розробка програмної платформи для моделювання, автоматизованого бенчмаркінгу та оркестрації обчислень у гібридних квантово-класичних периферійно-хмарних системах, що забезпечує високу точність оцінки наскрізних затримок та стійкості алгоритмів при збереженні обчислювальної та ресурсної ефективності процесу профілювання.

Об’єктом дослідження є процеси виконання гібридного програмного забезпечення та оцінки продуктивності гетерогенних квантово-класичних обчислювальних середовищ.

Предметом дослідження є архітектура кросплатформних бенчмаркінгових каркасів, алгоритми математичного оцінювання латентності транспіляції, методи операційного розрізнення вимірювань фон Неймана та модульні програмні засоби інтеграції хмарних квантових сервісів.

Для досягнення поставленої мети в роботі сформульовано та вирішено такі завдання:

- Проаналізувати проблему оцінки ефективності розподіленого програмного забезпечення в гібридних середовищах ери NISQ та виявити причини обмеженої масштабованості існуючих методів апаратного бенчмаркінгу.
- Розробити математичну та програмну модель наскрізної латентності (Latency Scores) як логічної комбінації мережевих, серверних часових характеристик та рівнів компіляторної оптимізації транспіляції.
- Спроектувати та реалізувати алгоритм автоматизованого вибору оптимального рівня транспіляції під задані обмеження точності (fidelity constraints) для мінімізації наскрізних затримок обчислень.

- Створити універсальний програмний каркас (фреймворк) для комплексного бенчмаркінгу, що об'єднує готові CLI-інструменти для низькорівневого тестування квантових вимірювань та програмні модулі виконання канонічних алгоритмів (Шора, Гровера, квантових блукань) в межах єдиної моделі взаємодії з хмарними провайдерами IBM Quantum та Amazon Braket.
- Провести експериментальну перевірку розробленого методу та програмного забезпечення через емуляцію мережевих затримок у віртуалізованому середовищі та порівняльний аналіз роботи симуляторів і реальних квантових процесорів із оцінкою ефективності алгоритмів коригування помилок зчитування.

Практичне значення отриманих результатів. У роботі розвинено архітектурний підхід, алгоритми та програмний інструментарій, які дозволяють будувати комплексні моделі оцінки продуктивності для аналізу та розгортання гібридного програмного забезпечення. Практична цінність рішення підтверджується тим, що запропоновані програмні інструменти (зокрема модулі інтеграції з хмарними API та інструменти низькорівневого тестування вимірювань) демонструють високу ефективність у задачах профілювання коду, успішно долаючи проблему ресурсомісткості та комбінаторного вибуху конфігурацій компіляції. Створена платформа дозволяє інтегрувати сучасні методи машинного навчання, збору статистичних даних та постпроцесингу для створення надійних систем автоматизованої оркестрації обчислень, що є критично важливим для розробки сучасного високопродуктивного інтелектуального програмного забезпечення.

1. АНАЛІЗ АКТУАЛЬНИХ ПРОБЛЕМ І ФОРМУЛЮВАННЯ ЗАВДАНЬ РОБОТИ

Хоча квантові обчислення все ще перебувають на ранніх стадіях свого розвитку, вони широко визнані як трансформаційний крок в обчислювальних технологіях, що пропонує можливості, які значно перевищують потенціал класичних систем. На відміну від класичних комп'ютерів, які існують у взаємовиключних станах (0 або 1), квантові комп'ютери використовують квантові біти, або кубіти, які можуть перебувати в суперпозиції станів, представляючи та обробляючи кілька можливостей одночасно [1]. Ця фундаментальна відмінність дозволяє квантовим системам набагато ефективнішим чином вирішувати складні проблеми, особливо ті, що пов'язані з багатовимірними просторами пошуку та імовірнісним виведенням.

1.1 Аналіз проблематики предметної області й постановка задач кваліфікаційної роботи

Суперпозиція є визначальною особливістю квантових систем, яка дозволяє кожному кубіту одночасно займати кілька потенційних станів, кожен з яких пов'язаний з певною ймовірністю спостереження. Проте, хоча кубіти можуть існувати в численних станах суперпозиції, вони колапсують до одного з двох можливих фізичних станів, представлених як $|0\rangle$ або $|1\rangle$, під час здійснення вимірювання або навіть через випадкову взаємодію з навколишнім середовищем [2]. Ця імовірнісна природа, на відміну від детермінованої поведінки класичних бітів, дозволяє квантовим комп'ютерам паралельно обробляти величезні обсяги інформації [3]. Суперпозиція також є важливою для квантової заплутаності — ще одного фундаментального принципу, за якого два або більше кубітів корелюють так, що стан одного безпосередньо визначає стан іншого, незалежно від їхньої фізичної відстані. Заплутані стани — це особливі зв'язки між двома

квантовими частинками, незалежно від їхнього просторового розділення. Наприклад, у випадку стану Белла:

$$|\Phi^+\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}, \quad (1.1)$$

якщо одна частинка перебуває у стані «0», інша також обов'язково буде у стані «0», а якщо перша перебуває у стані «1», інша так само опиниться в стані «1». Ця потужна властивість лежить в основі низки критично важливих застосунків, таких як квантова телепортація, безпечний зв'язок та розподілені квантові обчислення [4, 5].

Разом суперпозиція та заплутаність дозволяють квантовим системам перевершувати класичні архітектури, забезпечуючи перспективні прориви у таких стратегічних галузях, як криптографія, оптимізація, машинне навчання та квантові мережі. Сьогодні квантові пристрої стрімко завойовують ринок, пропонуючи широкий вибір устаткування на базі різних архітектур та супровідних програмних рішень. Серед постачальників обладнання, що пропонують публічний доступ до своїх вентильних (gate-based) пристроїв, можна виділити Rigetti, IBM, Oxford Quantum Group, IonQ та Xanadu [6-9]. Інші вендори пропонують пристрої, що працюють в інших парадигмах. Зокрема, варто згадати компанію D-Wave та їхні квантові відпалювачі (quantum annealers), або пристрої від QuEra, які функціонують на основі нейтральних атомів. Більшість вендорів надають власний програмний стек та інтерфейс прикладного програмування (API) для доступу до своїх пристроїв. Наприклад, комп'ютери Rigetti доступні через їхній Forest SDK та бібліотеку PyQuil, а доступ до комп'ютерів IBM Q можна отримати через Qiskit або веб-інтерфейс IBM Quantum Experience.

Деякі хмарні сервіси, такі як Amazon Braket, пропонують доступ до кількох квантових пристроїв під єдиним уніфікованим API [10]. Крім того, на вищих рівнях абстракції існує кілька бібліотек та фреймворків, здатних інтегруватися з обчислювальними потужностями багатьох постачальників обладнання одночасно. Прикладами таких інструментів є Qiskit від IBM Q, Orquestra від Zapata Computing, XACC, а також CUDA Quantum від компанії NVIDIA. Усі доступні на сьогоднішній

день квантові комп’ютери змушені класифікуватися як пристрої NISQ (Noisy Intermediate-Scale Quantum) — шумове квантове устаткування проміжного масштабу. Термін NISQ позначає квантові комп’ютери проміжного розміру, що очікуються в найближчі кілька років і зазвичай містять від 50 до кількох сотень кубітів. Значимість досягнення позначки у 50 кубітів полягає в перевершенні можливостей моделювання та симуляції сучасних цифрових суперкомп’ютерів. Слово «шумовий» (Noisy) підкреслює недосконалий контроль над кубітами, де технологічний шум накладає суттєві обмеження на досяжні результати квантових пристроїв у короткостроковій перспективі. Поточне квантове апаратне забезпечення суттєво обмежене такими інженерними викликами, як короткий час когерентності, помилки квантових вентилів і висока чутливість до шуму навколишнього середовища. Ці обмеження стають ще більш вираженими та критичними при інтеграції квантових процесорів (QPU) у розподілені периферійно-хмарні (edge-cloud) середовища, де додатково необхідно враховувати та систематично аналізувати такі фактори, як мережева затримка, синхронізація процесів та обмеження пропускної здатності каналів зв’язку.

Актуальність теми роботи. Незважаючи на зазначені апаратні та системні виклики, дослідження гібридних квантово-класичних периферійно-хмарних систем наразі стрімко набирає обертів. Цей технологічний прогрес активно підтримується зростаючою доступністю як симульованих, так і реальних квантових платформ, таких як IBM Quantum Lab та Amazon Braket [10-16]. Для забезпечення ефективної інтеграції та використання QPU у таких нових розподілених середовищах критично необхідним є створення надійного, стандартизованого фреймворку для бенчмаркінгу. Такий фреймворк покликаний забезпечити послідовний та уніфікований спосіб вимірювання, порівняння та оцінки продуктивності різних обчислювальних платформ, допомагаючи дослідникам та розробникам програмного забезпечення обирати найбільш підходящі варіанти для конкретних прикладних застосунків.

Останні досягнення в галузі квантових обчислень прискорили розробку методологій бенчмаркінгу (зокрема, методів рандомізованого тестування або

оцінок на основі квантового об’єму), які оцінюють апаратне забезпечення на основі послідовних показників: кількості кубітів, точності вентилів, часу когерентності та глибини квантової схеми, що виводяться під час виконання репрезентативних алгоритмів. Проте в специфічних контекстах периферійних хмар настільки ж критичними стають додаткові інженерні метрики, такі як затримка транспіляції, загальний час виконання та накладні витрати на комунікацію. Більшість цих метрик залежать від конкретного заліза та параметрів мережі, проте їх можна систематично охарактеризувати за допомогою визначення стандартизованого середовища тестування з контрольованими умовами та статистичним аналізом. Поряд із цим, існує гостра потреба в альтернативних методологіях тестування вентильних пристроїв з простою операційною інтерпретацією — наприклад, шляхом оцінки здатності пристрою вгадувати, яке саме з двох відомих квантових вимірювань фон Неймана було виконано в ході експерименту. Розробка відкритого програмного забезпечення, що об’єднує ці підходи та надає зручний інтерфейс командного рядка (CLI) з підтримкою Qiskit та Amazon Braket, є вкрай актуальним завданням для інженерії програмного забезпечення.

Метою роботи є розробка комплексного інноваційного програмного фреймворку та відкритого інструментарію, адаптованого для систематичного бенчмаркінгу і оцінки продуктивності гібридних квантово-класичних периферійно-хмарних середовищ та вентильних квантових обчислювальних пристроїв.

Об’єктом дослідження є процеси обчислень, транспіляції, мережевої комунікації та оцінки продуктивності в гібридних розподілених квантово-класичних системах.

Предметом дослідження є архітектурні шаблони, програмні інтерфейси (API), метрики ефективності та набір канонічних квантових алгоритмів і методів квантових вимірювань для всебічного бенчмаркінгу NISQ-пристроїв.

Для досягнення поставленої мети в роботі сформульовано та вирішено такі завдання:

1. Проаналізувати архітектурні особливості сучасного ринку квантових пристроїв, провайдерів доступу (Rigetti, IBM, Oxford Quantum Group, IonQ, Xanadu, D-Wave, QuEra) та наявних програмних стеків і бібліотек інтеграції.

2. Проектувати та реалізувати новий архітектурний підхід і фреймворк бенчмаркінгу для оцінки функціонування квантових систем у межах периферійно-хмарних мереж із використанням реальних метрик продуктивності.

3. Інтегрувати у фреймворк набір усталених канонічних квантових алгоритмів — алгоритм Шора, алгоритм Гровера та алгоритм квантових блукань (Quantum Walks), відомих своєю високою обчислювальною інтенсивністю та класичною нерозв'язністю, для оцінки платформ під різними обчислювальними навантаженнями.

4. Розробити та імплементувати допоміжну комунікаційну модель у віртуалізованому середовищі для точної емуляції реальних мережевих затримок, забезпечуючи всебічну оцінку комунікаційної продуктивності та накладних витрат.

5. Провести комплексні лонгітюдні (тривалі) експерименти на симульованих та реальних квантових апаратних платформах, включаючи середовища IBM Quantum Lab та Amazon Braket, з метою виявлення специфічних трендів платформ та системних проблем інтеграції квантових технологій у реальні практичні завдання.

6. Запропонувати стратегії оптимізації продуктивності гібридної системи, отримані на основі прикладного вартісного аналізу (value analysis), що дозволяють здійснювати обґрунтоване налаштування параметрів системи для підвищення її загальної ефективності. Розробити відкриту програмну бібліотеку мовою Python та утиліту командного рядка (CLI) для реалізації методу бенчмаркінгу вентильних пристроїв через перевірку їхньої спроможності розрізняти два вимірювання фон Неймана, забезпечивши підтримку як зумовлених сімейств вимірювань, так і визначених користувачем сценаріїв через інтерфейси Qiskit та Amazon Braket.

Наукова новизна та практичне значення отриманих результатів. Шляхом побудови стандартизованого, відтворюваного та відкритого підходу до бенчмаркінгу, дане дослідження створює фундаментальний програмний

інструментарій для всебічної оцінки квантових систем у розподілених обчислювальних середовищах. Створене програмне забезпечення дозволяє розробникам систем автоматизовано збирати метрики точності вентилів, оверхеду транспіляції та латентності мережі, що безпосередньо підтримує практичний прогрес у напрямку інтеграції квантових обчислювачів у хмарні сценарії та наближає досягнення реальної квантової переваги.

1.2 Основні підходи та конкурентні рішення до реалізації бенчмаркінгу квантових комп'ютерів.

На сучасному етапі розвитку обчислювальної техніки спостерігається стрімке поширення квантових пристроїв, що характеризується розмаїттям апаратних архітектур та відповідних програмних рішень. Серед провідних розробників, які надають публічний доступ до своїх квантових обчислювальних систем на базі вентилів, варто відзначити такі компанії, як Rigetti [1], IBM [2], Oxford Quantum Group [3], IonQ [4] та Xanadu [5]. Поряд із цим існують альтернативні парадигми квантових обчислень, зокрема квантові відпалювачі від компанії D-Wave [6] та пристрої на основі нейтральних атомів від QuEra [7]. Для забезпечення взаємодії з апаратним забезпеченням більшість розробників пропонують власні програмні стеки та інтерфейси прикладного програмування (API). Наприклад, екосистема Rigetti підтримується через Forest SDK [8] та бібліотеку PyQuil [9], тоді як доступ до ресурсів IBM Q [2] реалізується за допомогою фреймворку Qiskit [10] або через веб-інтерфейс IBM Quantum Experience [11]. Розподілені хмарні платформи, зокрема Amazon Braket [12], забезпечують доступ до гетерогенних квантових пристроїв через єдиний уніфікований API. Крім того, існує низка програмних фреймворків, розроблених для кросплатформної інтеграції з обладнанням від різних постачальників, таких як Qiskit від IBM Q, Orquestra від Zapata Computing [13], XACC [14] та CUDA Quantum від NVIDIA [15].

Переважна більшість сучасних квантових комп'ютерів належить до класу NISQ (Noisy Intermediate-Scale Quantum) — шумових квантових пристроїв проміжного масштабу. Зазначена категорія охоплює квантові системи, що містять від 50 до кількох сотень кубітів, поява яких прогнозується найближчими роками. Досягнення межі у 50 кубітів є критично важливим етапом, оскільки це дозволяє перевищити межі можливостей симуляції класичних цифрових суперкомп'ютерів. Проте наявність неконтрольованого технологічного шуму суттєво обмежує точність маніпуляцій із кубітами, що в короткостроковій перспективі звужує спектр досяжних результатів та обчислювальних можливостей таких пристроїв [16, 17]. У зв'язку з цим постає фундаментальне науково-практичне завдання: визначити межі здатності пристроїв класу NISQ до виконання значущих обчислень. Для вирішення цього завдання критично необхідною є розробка формалізованих методологій їхнього бенчмаркінгу. У контексті вентильних квантових комп'ютерів, які є предметом даного дослідження, вже існує низка підходів, зокрема рандомізоване тестування [18–22] та метрики на основі оцінки квантового об'єму [23–25].

У межах даної кваліфікаційної роботи пропонується та досліджується альтернативний підхід до бенчмаркінгу вентильних пристроїв, який базується на строгій операційній інтерпретації. Запропонований метод дозволяє оцінити здатність конкретного квантового пристрою до розрізнення двох відомих вимірювань фон Неймана в процесі виконання експериментального сценарію. Практичну реалізацію даного підходу виконано у вигляді відкритого програмного забезпечення — бібліотеки мовою Python. Розроблене програмне рішення підтримує будь-які пристрої, доступні через екосистему Qiskit, що забезпечує його сумісність із провайдерами квантових послуг рівня IBM Q та Amazon Braket. Додатково у межах програмного пакета створено інструментарій командного рядка (CLI) для автоматизації процесів тестування та виконання найпоширеніших сценаріїв бенчмаркінгу.

Очікувано, що розроблене у межах даної роботи програмне рішення не є єдиним пакетом для бенчмаркінгу вентильних пристроїв. Незважаючи на те, що

запропонований підхід має значні переваги над іншими техніками тестування, задля забезпечення повноти дослідження у цьому розділі проводиться огляд та аналіз деяких із наявних на сьогодні аналогічних програмних засобів.

Ймовірно, найпростішим методом бенчмаркінгу, який можна розробити, є безпосереднє виконання відомих канонічних алгоритмів із подальшим порівнянням отриманих результатів з очікуваними (еталонними). Аналіз частоти отримання правильних вихідних даних або відхилення розподілу фактичних результатів від очікуваних забезпечує базову метрику продуктивності конкретного обчислювального пристрою. Такі програмні бібліотеки, як Munich Quantum Toolkit (MQT) [26, 27] або SupermarQ [28, 29], містять набори тестів, що використовують різноманітні алгоритми, зокрема алгоритми Шора та Гровера.

Попри свою інтуїтивну зрозумілість та легкість інтерпретації, подібні бенчмарки мають певні недоліки. Найважливішим із них є те, що вони оцінюють корисність квантового пристрою виключно в контексті виконання одного дуже специфічного алгоритму, через що результати такого тестування вкрай складно екстраполювати на інші класи алгоритмів та прикладних задач. Наприклад, нездатність пристрою стабільно знаходити факторизації за допомогою алгоритму Шора не дає жодної інформації про його потенційну ефективність у контексті варіаційних квантових алгоритмів (Variational Quantum Algorithms). Альтернативним підходом до бенчмаркінгу квантових комп'ютерів є метод рандомізованого тестування (randomized benchmarking).

У межах цього підходу здійснюється вибірка квантових схем для виконання із заздалегідь визначеного набору вентилів (наприклад, із групи Кліффорда), після чого оцінюється ступінь відхилення вихідного розподілу, отриманого на реальному пристрої під час виконання цих схем, від ідеального теоретичного розподілу. Крім того, поширеною практикою є конкатенація випадково обраних схем із їхніми оберненими аналогами (що в результаті має давати тотожну схему, тобто identity circuit) та подальше виконання цих з'єднаних ланцюгів на фізичному пристрої. До програмних бібліотек, що реалізують цей підхід, належать фреймворки Qiskit [30] та PyQuil [31]. Іншим, не менш популярним методом бенчмаркінгу є квантова

томографія (quantum tomography) [32]. Практичні приклади реалізації методів квантової томографії для тестування пристроїв класу NISQ наведено в роботі [33].

У дослідженні [34] автори здійснили оцінку продуктивності низки обчислювальних машин IBM Q, застосувавши сім різних еталонних тестів (бенчмарків), які враховували показники помилок та загальний час виконання алгоритмів. Програмний пакет QASMBench [35] є одним із перших комплексних наборів тестів, орієнтованих на оцінку пристроїв класу NISQ із використанням квантових застосунків із широкого спектра предметних областей, що переважно базується на підході оцінки точності виконання (fidelity estimation). Представлений у згаданій роботі бенчмарк дозволяє проводити порівняльний аналіз точності виконання обчислень на машинах екосистеми IBM Q, квантових процесорах (QPU) IonQ та обчислювальній системі Rigetti Aspen M-1.

Іншою важливою метрикою, що широко використовується для бенчмаркінгу пристроїв NISQ, є квантовий об'єм (quantum volume). Квантовий об'єм кількісно характеризує загальну здатність квантового пристрою до вирішення складних обчислювальних задач. Цей інтегральний показник враховує комплекс апаратних факторів, зокрема таких як кількість кубітів, топологія їхньої зв'язності (connectivity) та рівень помилок вимірювання. Бібліотека Qiskit надає розробникам готовий інструментарій для вимірювання квантового об'єму пристрою за допомогою спеціалізованого програмного модуля qiskit.ignis.verification.quantum_volume. Існують також інші альтернативні програмні реалізації методів розрахунку квантового об'єму, детальний опис яких наведено, зокрема, у праці [36].

Окремої уваги заслуговує метод бенчмаркінгу на основі перехресної ентропії (cross-entropy benchmarking) [37], який був успішно застосований для валідації експериментів із досягнення квантової переваги на 53-кубітному процесорі Sycamore [38]. У межах цього підходу якість алгоритму, імплементованого на QPU, вимірюється шляхом обчислення перехресної ентропії послідовностей бітів (bit-strings), які фактично зчитуються (семплюються) з квантового процесора, у порівнянні з ідеальними теоретично розрахованими послідовностями.

Підсумовуючи огляд існуючих підходів, варто зазначити, що наразі у науковій та інженерній спільнотах триває активна робота, спрямована на комплексну стандартизацію процесів бенчмаркінгу квантових комп'ютерів. Наприклад, у дослідженні [39] автори представляють концепцію розробки уніфікованого набору тестів, який повністю базується на вимірюванні та моніторингу комплексу стандартизованих ключових показників ефективності (KPI).

1.3 Постановка проблеми. Попередня розробка архітектури та підхід до схеми дискримінації. Вимірювання фон Неймана. Схема дискримінації

У даному розділі наведено детальний опис функціонування процесу бенчмаркінгу, реалізованого в розробленому програмному забезпеченні. Дослідження розпочинається з викладу необхідних математичних основ. Після цього представлено загальну форму схеми дискримінації, що застосовується у запропонованій системі, а також наведено практичні аспекти її імплементації з огляду на технічні обмеження сучасних пристроїв класу NISQ.

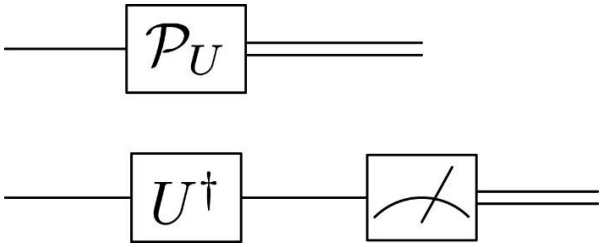

Рисунок 1.1 – Реалізація вимірювання фон Неймана за допомогою вимірювання в обчислювальному базисі.

Верхня схема демонструє символічне представлення вимірювання фон Неймана $\mathcal{P}_U$. Нижня, еквівалентна схема, відображає його декомпозицію на зміну базису з подальшим вимірюванням у Z-базисі.

Вимірювання фон Неймана $\mathcal{P}$ являє собою сукупність проєкторів першого рангу $\{|u_0\rangle\langle u_0|,\ldots,|u_{d-1}\rangle\langle u_{d-1}|\}$, які називаються ефектами та в сумі дають тотожний

(одиничний) оператор, тобто $\sum_{i=0}^{d-1} |u_i\rangle\langle u_i| = 1$. Якщо U є унітарною матрицею розмірності d, можна побудувати вимірювання фон Неймана $\mathcal{P}_U$, використовуючи проєктори на її стовпці. У такому випадку вважається, що вимірювання $\mathcal{P}_U$ описується матрицею U. Як правило, пристрої класу NISQ здатні виконувати вимірювання виключно в стандартному обчислювальному Z-базисі, тобто за умови, що $U = 1$. Для імплементації довільного вимірювання фон Неймана $\mathcal{P}_U$ необхідно спочатку застосувати оператор $U^\dagger$ до досліджуваної системи, після чого виконати вимірювання у Z-базисі. Цей процес можна розглядати як зміну базису, що виконується безпосередньо перед вимірюванням у стандартному обчислювальному базисі, як показано на Рис. 1.1.

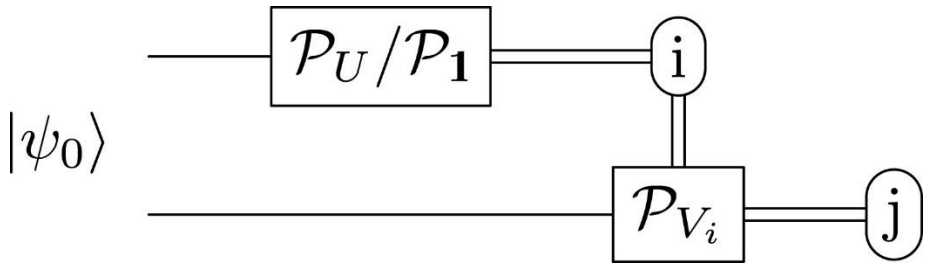

Рисунок 1.2 – Теоретична схема дискримінації між вимірюваннями фон Неймана $\mathcal{P}_U$ та $\mathcal{P}$.

Бенчмарки у розробленому програмному забезпеченні функціонують шляхом експериментального визначення ймовірності правильного розрізнення (дискримінації) між двома вимірюваннями фон Неймана на досліджуваному пристрої та порівняння отриманого результату з ідеальними теоретичними передбаченнями. Без втрати загальності ми розглядаємо задачу розрізнення між вимірюванням одного кубіта $\mathcal{P}$, що виконується в обчислювальному Z-базисі, та альтернативним вимірюванням $\mathcal{P}_U$, що виконується в базисі U. Схема дискримінації, представлена на Рис. 2, потребує використання допоміжного кубіта (auxiliary qubit).

Спочатку об’єднана система готується в деякому стані $|\psi_0\rangle$. Потім одне з вимірювань, або $\mathcal{P}_U$, або $\mathcal{P}$, виконується над першою частиною системи. На основі отриманого результату i ми обираємо інше бінарне вимірювання $\mathcal{P}_{V_i}$ і виконуємо його над другим кубітом, отримуючи результат j. Зрештою, якщо $j=0$, ми вважаємо, що було виконано вимірювання $\mathcal{P}_U$, в іншому випадку — що це було вимірювання $\mathcal{P}$. Слід зауважити, що описана вище схема дискримінації може працювати незалежно від розмірності унітарної матриці U. Основна відмінність полягає в розмірності допоміжної системи, яка загалом може бути більшою за два. Ця розмірність залежить від рангу Шмідта (Schmidt rank) оптимального вхідного стану. Проте для більшості схем розрізнення ранг Шмідта не перевищує двох, і, отже, допоміжна система також є кубітом (детальніше див. [41]). Зауважимо, що фінальне вимірювання $\mathcal{P}_{V_i}$ завжди є бінарним, незалежно від розмірності допоміжної системи. Очевидно, що для отримання надійної оцінки базового розподілу ймовірностей нам необхідно повторити цю процедуру багато разів для обох вимірювань. У запропонованій бібліотеці припускається, що експеримент повторюється однакову кількість разів як для $\mathcal{P}_U$, так і для $\mathcal{P}$.

У принципі, наша схема дискримінації може бути використана з будь-яким вибором стану $|\psi_0\rangle$ та фінальних вимірювань $\mathcal{P}_{V_i}$. Однак ми стверджуємо, що найкраще обирати ці компоненти таким чином, щоб вони максимізували ймовірність правильного розрізнення. Щоб переконатися в цьому, припустимо, що деякий вибір $|\psi_0\rangle, \mathcal{P}_{V_0}, \mathcal{P}_{V_1}$ дозволяє правильно розрізняти два вимірювання з ймовірністю, яка дорівнює одиниці, тобто на ідеальному квантовому комп’ютері ви завжди зробите правильний вибір. Тоді на реальному фізичному обладнанні ми можемо отримати будь-яке емпіричне значення в діапазоні $[\frac{1}{2},1]$. З іншого боку, якщо ми оберемо компоненти нашої схеми так, щоб ймовірність успішного

розрізнення становила $\frac{3}{5}$, можливий діапазон емпірично отриманих ймовірностей становитиме лише $[\frac{1}{2},\frac{3}{5}]$. Отже, у другому випадку розбіжність між теоретичними та емпіричними результатами буде менш вираженою.

1.4 Реалізація схеми дискримінації на реальних пристроях NISQ

Сучасні пристрої класу NISQ не здатні виконувати умовні вимірювання (conditional measurements), що є найбільшою перешкодою для реалізації нашої схеми на реальному фізичному обладнанні. Однак ми обходимо цю проблему шляхом незначного коригування нашої схеми таким чином, щоб вона використовувала лише ті компоненти, які доступні на поточних пристроях. Для цього ми застосовуємо два можливі варіанти: використання постселекції (postselection) або прямої суми $V_0^\dagger \oplus V_1^\dagger$.

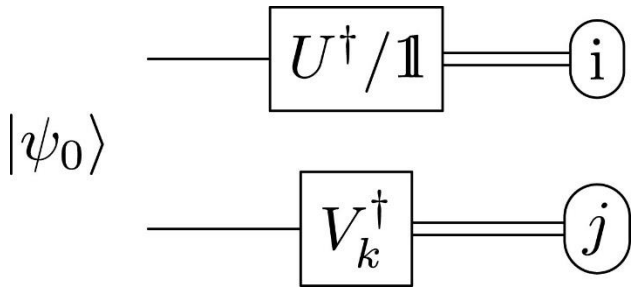

Рисунок 1.3 – Схематичне представлення установки для розрізнення вимірювань $\mathcal{P}_U$ та $\mathcal{P}$ за допомогою підходу постселекції. У схемі постселекції такі квантові схеми запускаються для обох значень $k=0,1$, а результати для тих випадків, коли виникає невідповідність між k та i, відкидаються.

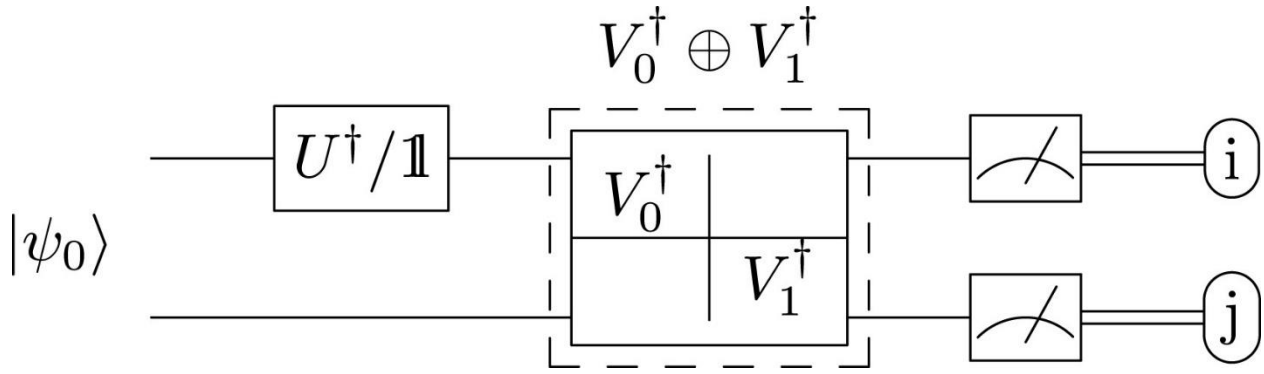

Рисунок 1.4 – Схематичне представлення установки для розрізнення вимірювань $\mathcal{P}_U$ та $\mathcal{P}$ з використанням прямої суми $V_0^\dagger \oplus V_1^\dagger$.

Схема 1 (Постселекція). Перша ідея використовує схему постселекції. В оригінальній схемі ми вимірюємо перший кубіт і лише потім визначаємо, яке вимірювання слід виконати над другим. Замість того, щоб робити цей вибір, ми можемо запустити дві схеми: одну з $\mathcal{P}_{V_0}$, а іншу з $\mathcal{P}_{V_1}$, і виміряти обидва кубіти. Потім ми відкидаємо результати тієї схеми, для якої мітка i не збігається з міткою вимірювання k. Отже, схема постселекції виглядає так, як показано на Рис. 1.3. Для виконання бенчмарку необхідно запустити кілька копій схеми постселекції як з $\mathcal{P}_U$, так і з $\mathcal{P}$. Кожна схема має бути запущена в обох варіантах: один із фінальним вимірюванням $\mathcal{P}_{V_0}$, а другий — з фінальним вимірюванням $\mathcal{P}_{V_1}$. Таким чином, експерименти можна згрупувати в класи, що ідентифікуються кортежами вигляду (Q,k,i,j), де $Q \in \{\mathcal{P}_U, \mathcal{P}\}$ позначає обране вимірювання, $k \in \{0,1\}$ визначає використане фінальне вимірювання, а $i \in \{0,1\}$ та $j \in \{0,1\}$ є мітками результатів, як показано на рис. 3. Потім ми відкидаємо всі експерименти, для яких $i \neq k$. Отже, загальна кількість валідних експериментів становить:

$$N_{total} = \#\{(Q,k,i,j) : k = i\}. \tag{1.2}$$

Нарешті, ми підраховуємо валідні експерименти, які призвели до успішного розрізнення (дискримінації). Якщо ми обрали $\mathcal{P}_U$, то ми робимо правильне

припущення тоді й лише тоді, коли $j=0$. Аналогічно, для $\mathcal{P}$, ми робимо правильне припущення тоді й лише тоді, коли $j=1$. Якщо ми визначимо:

$$N_{\mathcal{P}_U} = \#\{(Q,k,i,j) : Q = \mathcal{P}_U, k = i, j = 0\}, \tag{1.3}$$

$$N_{\mathcal{P}} = \#\{(Q,k,i,j) : Q = \mathcal{P}, k = i, j = 1\}, \tag{1.4}$$

тоді емпірична ймовірність успіху може бути обчислена як

$$p_{\text{succ}}(\mathcal{P}_U, \mathcal{P}) = \frac{N_{\mathcal{P}_U} + N_{\mathcal{P}}}{N_{\text{total}}}. \tag{1.5}$$

Величина p_{succ} — це показник, який повідомляється користувачеві як результат бенчмарку.

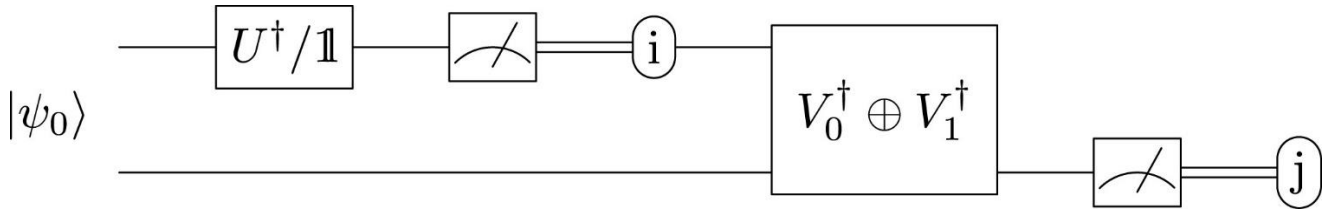

Рисунок 1.5 – Схематичне представлення установки для розрізнення вимірювань $\mathcal{P}_U$ та $\mathcal{P}$ з використанням прямої суми $V_0^\dagger \oplus V_1^\dagger$.

Схема 2 (Пряма сума). Друга ідея використовує реалізацію на основі прямої суми $V_0^\dagger \oplus V_1^\dagger$. Тут, замість виконання умовного вимірювання $\mathcal{P}_{V_k}$, де $k \in \{0,1\}$, ми запускаємо схеми, представлені на Рис. 1.4. Можна побачити, чому така схема є еквівалентною оригінальній схемі дискримінації. Якщо ми перепишемо блочно-діагональну матрицю $V_0^\dagger \oplus V_1^\dagger$ наступним чином:

$$V_0^\dagger \oplus V_1^\dagger = |0\rangle\langle 0| \otimes V_0^\dagger + |1\rangle\langle 1| \otimes V_1^\dagger, \tag{1.6}$$

ми побачимо, що пряма сума в рівнянні (1.5) комутує з вимірюванням над першим кубітом. Завдяки цьому ми можемо змінити порядок операцій й отримати схему, зображену на Рис. 1.5. Тепер, залежно від результату i, один із доданків у рівнянні (5) зникає, і в підсумку ми виконуємо точно такі ж операції, як і в оригінальній

схемі. У цій схемі експеримент можна охарактеризувати набором (Q,i,j), де $Q \in \{\mathcal{P}_U, \mathcal{P}\}$, а $i,j \in \{0,1\}$ є мітками вихідних результатів. Кількість успішних спроб для U та відповідно можна записати як:

$$N_{\mathcal{P}_U} = \#\{(Q,i,j) : Q = \mathcal{P}_U, j = 0\}, \tag{1.7}$$

$$N_{\mathcal{P}} = \#\{(Q,i,j) : Q = \mathcal{P}, j = 1\}. \tag{1.8}$$

Тоді ймовірність правильного розрізнення (дискримінації) між $\mathcal{P}_U$ та $\mathcal{P}$ задається виразом:

$$p_{\text{succ}} = \frac{N_{\mathcal{P}_U} + N_{\mathcal{P}}}{N_{\text{total}}}, \tag{1.9}$$

де N_{total} — це загальна кількість спроб.

Порівняно з цими підходами, наш підхід забезпечує дуже просту операційну інтерпретацію схеми та її результатів. Це особливо корисно для новачків у цій галузі, яких можуть відлякати складніші підходи. Іншою перевагою, особливо для досвідчених користувачів, є можливість контролювати ресурси, що використовуються під час бенчмаркінгу. Тут ми можемо розглядати такі ресурси, як заплутаність (entanglement) або когерентність (coherence). Зрештою, цільовий показник (figure of merit), який ми хочемо обчислити і з яким порівнюємо результати, отримати досить просто. Головним недоліком нашого підходу є експоненціальна кількість квантових схем, які необхідно враховувати.

2. ОСНОВНІ ПАРАДИГМИ БЕНЧМАРКІНГУ КВАНТОВИХ ОБЧИСЛЮВАЛЬНИХ СИТЕМ. ПРОЄКТУВАННЯ АРХІТЕКТУРИ ПРОГРАМНОЇ СИСТЕМИ

Квантовомеханічні явища за своєю суттю є складними для розуміння через їхній відхід від класичної інтуїції та непридатність повсякденного макроскопічного досвіду. Квантова механіка являє собою математичну основу, що характеризується набором фундаментальних аксіом, з яких логічно виводиться поведінка квантових систем [17].

2.1 Розгортання та концептуальна основа програмної системи. Квантові обчислення на основі вентилів. Квантові пристрої в гібридних периферійних архітектурах

Простори квантових станів разом із трансформаціями, що діють на них, зазвичай формулюються з використанням лінійно-алгебраїчних структур, таких як вектори, матриці та бра-кет нотація Дірака. Ці вектори станів представляють частинки, такі як електрони або поляризовані фотони [18], і містяться в n-вимірному комплексному гільбертовому просторі [18]. Сьогодні більшість фізиків використовують нотацію Дірака через її лаконічність та ясність [19]. Наприклад, вектор квантового стану позначається як кет, $|q\rangle$, а його ермітово-спряжений (дуальний вектор) записується як бра, $\langle q|$. Типовий кет $|q\rangle$ часто є суперпозицією двох базисних векторів, $|0\rangle$ та $|1\rangle$, з комплексними амплітудами ймовірності α та β. Математично це виражається як

$$|q\rangle = \alpha|0\rangle + \beta|1\rangle, \tag{2.1}$$

з умовою нормування

$$|\alpha|^2 + |\beta|^2 = 1. \tag{2.2}$$

У квантових обчисленнях кет $|q\rangle$ представляє кубіт. Ці квантові стани масштабуються експоненціально відповідно до фактора зростання 2^n, де n —

кількість кубітів. Це масштабування зазвичай представляється за допомогою тензорного добутку, що позначається символом $\otimes$ [20]. Як результат, квантові комп'ютери можуть представляти та обробляти інформацію способами, які далеко перевищують можливості класичних комп'ютерів.

Другий фундаментальний механізм називається квантовим паралелізмом [21]. Це стосується здатності квантового комп'ютера використовувати суперпозицію для одночасної обробки всіх можливих вхідних станів [17]. Суперпозиція дозволяє кубітам існувати в комбінації станів, що дає змогу квантовим комп'ютерам оцінювати кілька обчислювальних шляхів одночасно, тим самим прискорюючи певні обчислення. Алгоритм Шора є яскравим прикладом, який елегантно використовує квантовий паралелізм за допомогою квантового перетворення Фур'є [17]. Це докорінно відрізняється від класичного паралелізму, в якому операції виконуються над взаємовиключними станами [21].

Третім важливим аспектом квантових обчислень є заплутаність (entanglement), або заплутані квантові стани. Квантовий стан $|q\rangle$ називається заплутаним, якщо його не можна виразити як простий тензорний добуток двох індивідуальних станів кубітів. У контексті вимірювання дві частинки вважаються незаплутаними, якщо результат вимірювання однієї не впливає на результат іншої [16]. Натомість заплутані кубіти корелюють таким чином, що стан одного кубіта безпосередньо впливає на стан іншого, незалежно від відстані між ними. Завдяки цим трьом фундаментальним механізмам — суперпозиції, квантовому паралелізму та заплутаності — квантові обчислення здатні виконувати операції, які є нездійсненними для класичних комп'ютерів, і справлятися з масштабними обчислювальними задачами, що виходять за межі теоретичних можливостей класичних комп'ютерів, пропонуючи можливість вирішення проблем, які раніше вважалися нерозв'язними.

Квантові вентилі діють як оператори над кетами $|q\rangle$, які слугують операндами у квантових обчисленнях. Ці вентилі виконують лінійні перетворення над квантовими станами, і для збереження як ортогональності, так і норми векторів станів, ці перетворення повинні бути унітарними. Математично квантові

обчислення передбачають застосування серії унітарних перетворень до початкового квантового стану, який зазвичай позначається як $|q_0 q_1 \cdots q_n\rangle$. На відміну від класичних логічних вентилів, більшість квантових вентилів, таких як вентилі Адамара (Hadamard), Паулі (Pauli) та CNOT, за своєю суттю є унітарними, оскільки вони відповідають вимозі збереження ортогональності та оборотності [17]. Вентиль є унітарним, якщо його можна подати у вигляді унітарної матриці; його обернена матриця дорівнює його ермітово-спряженій матриці (conjugate transpose). Зрештою, у квантових обчисленнях квантовий стан вимірюється, що призводить до колапсу суперпозиції в класичний результат.

У гібридній квантово-класичній системі «периферія-хмара» (edge-cloud) квантові пристрої розподіляються по мережі відповідно до їхніх можливостей. Хоча розміщення є гнучким, пристрої на вищих рівнях (наприклад, у головній хмарі) зазвичай забезпечують більші обчислювальні або комунікаційні можливості, ніж пристрої на периферії або кінцевих точках [22]. Нижче наведено огляд ключових квантових пристроїв на цих рівнях:

1. Квантовий процесор (Quantum processing unit, QPU): QPU є основним обчислювальним компонентом квантового комп’ютера, який виконує квантові схеми на кубітах для вирішення складних завдань, що є непосильними для класичних комп’ютерів. Кількість фізичних кубітів у сучасних квантових комп’ютерах на основі вентилів, зокрема надпровідних кубітів (які використовують IBM та Google) та кубітів на захоплених іонах (які використовує Quantinuum), наразі становить менше ніж 1500. За такого сценарію алгоритми, які розподіляють робоче навантаження між класичними та квантовими ресурсами, є життєво необхідними для підвищення надійності та продуктивності.

2. Квантові комутатори забезпечують з’єднання каналів між пристроями у квантовій мережі для маршрутизації квантової інформації або розподілу заплутаностей. Квантовий комутатор може обробляти кілька квантових каналів одночасно [23].

3. Квантові ретранслятори (Quantum Repeaters, QR): ці пристрої збільшують відстань зв’язку, долаючи втрату сигналу без порушення принципу заборони

клонування (no-cloning principle), використовуючи такі методи, як телепортація, обмін заплутаністю (entanglement swapping) та виправлення помилок [24]. Квантові реле (Quantum relays) функціонують подібно, але не мають функції очищення заплутаності (entanglement purification) та квантової пам'яті; натомість вони покращують дальність передачі за рахунок підвищення відношення сигнал/шум на детекторах [25].

4. Квантова пам'ять: квантова пам'ять зберігає стани одиночних фотонів або заплутані стани протягом коротких періодів часу, не руйнуючи їхніх квантових властивостей. Вони сприяють гібридним периферійним обчисленням (hybrid edge computing), забезпечуючи синхронізацію операцій та зберігаючи заплутані стани доти, доки вузли не будуть вирівняні (aligned).

5. Квантові сенсори: використовують такі властивості, як суперпозиція та заплутаність, для досягнення вимірювальних можливостей, що значно перевищують можливості класичних сенсорів [26]. На периферії квантові сенсори забезпечують надточні вимірювання та покращують якість вхідних даних. Ці сенсори можуть надійно працювати в обмежених умовах, таких як поля бою та віддалені райони.

2.2. Архітектурні аспекти гібридних периферійно-хмарних систем. Реалізація квантового бенчмаркінгу

Гібридні квантово-класичні системи поєднують обчислювальну потужність квантових процесорів із надійністю, масштабованістю та гнучкістю класичних систем. Оскільки сучасні квантові пристрої належать до ери NISQ, яка характеризується обмеженою кількістю кубітів, коротким часом когерентності та значним шумом, класичні системи відіграють життєво важливу роль у компенсуванні їхніх обмежень. Вони забезпечують виконання таких важливих завдань, як попередня обробка даних (data preprocessing), ітеративний зворотний зв'язок та пом'якшення помилок (error mitigation). У цих гібридних середовищах квантові процесори зазвичай вирішують обчислювально складні завдання, такі як

оптимізація, моделювання та криптографія, тоді як класичні системи керують периферійними операціями. Синергія між двома парадигмами забезпечує можливості вирішення задач, яких жодна система не могла б досягти поодинці. Така співпраця є особливо актуальною для складних масштабних проблем, таких як логістика ланцюгів постачання або квантово-хімічне моделювання. Сучасні додатки за своєю природою є гібридними, вимагаючи як квантових, так і класичних ресурсів для ефективного функціонування. Класичні компоненти виконують попередню обробку для генерації схем підготовки станів та інтегрують їх у квантові алгоритми. Класичні компоненти також відповідають за подальшу обробку (postprocessing), наприклад, за виправлення помилок зчитування шляхом застосування методу розгортання (unfolding method) для обчислення незбуреного розподілу результатів зі збуреного виміряного розподілу. Наприклад, навіть такі квантові алгоритми, як алгоритм Шора, потребують класичної постобробки для завершення факторизації [27]. Для підтримки такого рівня інтеграції гібридна архітектура «периферія-хмара» (edge-cloud) розподіляє різноманітні обчислювальні ресурси — від IoT-пристроїв до доступного через хмару квантового обладнання — між периферійними та хмарними середовищами. Спеціальний шар оркестрації (orchestration layer) координує систему, динамічно призначаючи завдання класичним або квантовим компонентам на основі визначених політик або інтелектуальних механізмів оптимізації. На Рис 2.1 показано багаторівневу ієрархічну архітектуру гібридних систем, організовану у п’ять окремих рівнів:

1. Рівень кінцевих точок (End-point Layer): цей рівень включає сенсорні, виконавчі (actuating) та вимірювальні пристрої, розташовані поблизу фізичного середовища. Ці пристрої збирають великі обсяги необроблених даних, які необхідно обробити або передати для аналізу.

2. Периферійний рівень (Edge Layer): периферійний рівень включає локальні класичні обчислювальні вузли з помірною потужністю обробки та зберігання даних. Квантові пристрої також можуть бути інтегровані на периферії для підтримки гібридних обчислень, чутливих до затримок, ближче до джерела даних.

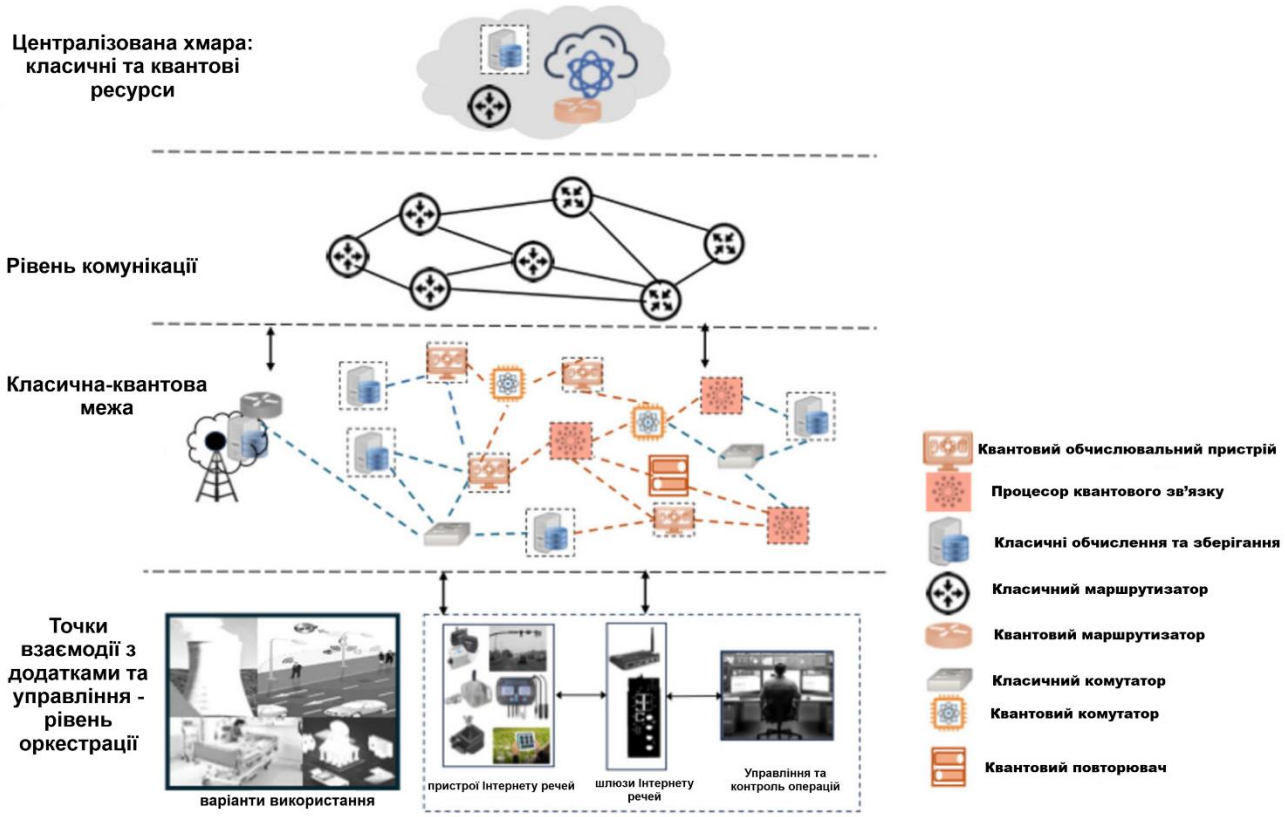

Рисунок 2.1 – Багаторівнева та ієрархічна архітектура гібридних квантово-класичних систем «край-хмара».

3. Рівень комунікації (Communication Layer): Цей рівень відповідає за взаємозв'язок між різними рівнями та пристроями, використовуючи класичні або квантові канали зв'язку залежно від типу даних. Він забезпечує безпечну передачу даних із низькою затримкою та високою пропускною здатністю по всій архітектурі.

4. Хмарний рівень (Cloud Layer): Хмарний рівень складається з потужних класичних центрів обробки даних та високопродуктивних квантових комп'ютерів (QPU), доступних через хмару. Він обробляє обчислювально інтенсивні завдання, що перевищують можливості периферійного рівня, такі як навчання масштабних моделей квантового машинного навчання або виконання складних квантових симуляцій.

5. Рівень оркестрації (Orchestration Layer): Виконуючи роль «мозку» гібридної архітектури, рівень оркестрації координує розподіл завдань і ресурсів між периферійним та хмарним рівнями. Він використовує інтелектуальні алгоритми оптимізації для визначення того, чи має робоче навантаження

виконуватися на класичному, чи на квантовому обладнанні, а також чи має воно оброблятися локально на периферії, чи віддалено в хмарі, на основі таких обмежень, як затримка, енергоспоживання та конфіденційність даних.

Вважається, що поточний стан квантових обчислень перебуває на етапі Noisy Intermediate-Scale Quantum (NISQ) — квантових технологій проміжного масштабу з наявністю шумів. Цей етап характеризується квантовими процесорами, здатними виконувати обчислення, непосильні для класичних комп'ютерів, але обмеженими через шум, скінченний час когерентності та невелику кількість кубітів. У цьому контексті квантовий бенчмаркінг набуває важливого значення, оскільки дослідники намагаються оцінювати та порівнювати квантові пристрої для визначення напрямків майбутніх розробок [28]. Доступність невеликих квантових комп'ютерів на основі вентилів, таких як ті, що надаються платформами Amazon Braket та IBM Quantum, прискорила розвиток практик квантового бенчмаркінгу. Ці платформи дозволяють дослідникам тестувати та порівнювати продуктивність різних квантових комп'ютерів шляхом запуску різноманітних квантових алгоритмів. Це допомагає їм побачити, наскільки добре пристрій працює порівняно з іншими пристроями або з очікуваними результатами теорії та класичних симуляцій [10].

Вентильний квантовий пристрій обробляє вхідні кубіти, виконуючи унітарні перетворення за допомогою структурованої послідовності вентильних операцій та вимірювань, що представляються у вигляді квантової схеми. Для оцінки надійності та продуктивності цих схем часто використовуються прості послідовності вентилів, наприклад, багаторазове застосування вентилів CNOT. На відміну від традиційних низькорівневих апаратних метрик, алгоритмічний бенчмаркінг став провідною методологією для оцінки продуктивності квантових обчислень. Цей підхід передбачає запуск повноцінних алгоритмів на квантових пристроях та оцінку їхньої коректності, часу виконання та стійкості. Він забезпечує більш релевантну для практичного застосування оцінку продуктивності системи, ніж оцінка точності (fidelities) окремих вентилів.

У класичній інженерії програмного забезпечення бенчмаркінг традиційно спирається на широкий консенсус, а не на загальноприйняте визначення. У широкому розумінні це процес запуску стандартизованих тестів для оцінки, за допомогою відповідних метрик, того, наскільки добре працює система, зазвичай шляхом порівняння її з іншими системами або еталонним стандартом.

Ця концептуальна база перейшла і в квантові обчислення, але квантовий бенчмаркінг несе із собою власний набір викликів та нюансів. Під час вимірювання продуктивності він має враховувати такі унікальні явища, як заплутаність, декогеренція та ймовірнісний характер вимірювань. Хоча терміни «квантові бенчмарки» та «квантові метрики» іноді використовуються як синоніми, вони є різними поняттями.

Квантова метрика — це кількісна міра продуктивності, призначена для оцінки конкретних аспектів роботи квантового пристрою або системи, таких як точність (fidelity) або рівень помилок. Бенчмарк, з іншого боку, є ширшим тестом, який може включати одну або кілька таких метрик. Загалом, квантовий бенчмаркінг передбачає тестування того, наскільки добре функціонує квантова обчислювальна система або її компоненти з використанням відомих алгоритмів та попередньо визначених метрик. Він допомагає дослідникам і розробникам оцінювати їхню точність, надійність та обчислювальні можливості в реальних або змодельованих умовах.

Бенчмаркінг також полегшує порівняння різних квантових платформ або відстеження покращення продуктивності з часом, що є вкрай важливим у міру того, як ця галузь продовжує розвиватися [29]. На відміну від бенчмаркінгу класичних обчислень, який часто використовує тестові набори загального призначення, такі як LINPACK та SPEC (Standard Performance Evaluation Corporation, Гейнсвілл, Вірджинія, США), квантовий бенчмаркінг за своєю природою є цілеспрямованим. Наприклад, його можна використовувати для оцінки продуктивності квантових процесорів у гібридних середовищах «периферія-хмара» (edge-cloud) або для оцінки конкретних апаратних компонентів, таких як кубіти, квантові вентилі, схеми та пристрої. Корисність результатів значною мірою залежить від того,

наскільки вони узгоджуються з початковим задумом тесту [30]. На поточному етапі розвитку квантових обчислень, коли системи все ще схильні до помилок і обмежені в масштабах, бенчмарки здебільшого слугують для вимірювання прогресу та виявлення «вузьких місць». Замість того, щоб оцінювати повноцінну обчислювальну корисність, більшість сучасних бенчмарків зосереджені на впливі шуму, частоті помилок та апаратних обмеженнях [31]. У нашій роботі ми зосереджуємось конкретно на бенчмаркінгу вентильних квантових комп'ютерів, де системи будуються з використанням добре зрозумілих компонентів, таких як кубіти, вентилі та схеми. Це дозволяє нам створювати бенчмарки, які є інтерпретованими та змістовними.

2.3. Види квантового бенчмаркінгу. Метрики квантового бенчмаркінгу. Бенчмаркінг у гібридних квантово-класичних гранично-хмарних системах

Бенчмаркінг у квантових обчисленнях, так само як і в класичних, спирається на вимірювані метрики продуктивності для порівняння систем із часом, між різними технологіями та за різних умов. Але бенчмаркінг квантових систем є набагато складнішим через притаманний їм шум, імовірнісну поведінку та апаратне різноманіття. Жоден окремий бенчмарк не може універсально оцінити квантове обладнання, тому часто використовується кілька метрик. Аналогічно, під час оцінки гібридних квантових обчислень у середовищі «периферія-хмара» (edge-cloud) ефективний бенчмаркінг має бути багатогранним. Він повинен охоплювати багато рівнів обчислювального стека за допомогою набору протоколів, спрямованих на конкретні аспекти продуктивності, релевантні для цільового застосування.

Щоб краще зрозуміти, яке місце займає наш метод бенчмаркінгу, корисно дослідити ключові рівні стека квантового бенчмаркінгу:

1. Бенчмаркінг на рівні кубітів (Qubit-Level Benchmarking) зосереджується на базових будівельних блоках квантових вентилів — кубітах. Кількість кубітів прямо

пов'язана з обчислювальною потужністю квантового процесора [32]. Ключовими метриками є час когерентності, який кількісно визначає тривалість, протягом якої кубіти зберігають свої квантові стани, та частотна характеристика, яка описує спектральну поведінку кубітів і їхню взаємодію з керівними сигналами.

2. Бенчмаркінг на рівні вентилів (Gate-Level Benchmarking) розглядає те, наскільки добре працюють окремі квантові вентилі. Він широко визнаний критично важливим компонентом будь-якого комплексного фреймворку квантового бенчмаркінгу. Поширені методи на цьому рівні включають:

- Рандомізований бенчмаркінг (Randomized benchmarking, RB): надає оцінку середньої точності (fidelity) вентиля, одночасно пом'якшуючи вплив помилок підготовки стану та вимірювання (SPAM). RB є масштабованим і менше піддається впливу шуму.
- Пряма оцінка точності (Direct Fidelity Estimation, DFE): оцінює точність вентиля за умови припущення про незначність помилок SPAM. DFE надає єдине значення від 0 до 1 як міру реалізації вентиля, де 1 означає ідеальну реалізацію.
- Квантова томографія (Quantum Set Tomography, QPT): пропонує повну характеризацію вентиля, але є ресурсомісткою і погано масштабується.

Оскільки квантові платформи покладаються як на високорівневі вентилі (що використовуються під час розробки алгоритмів), так і на нативні вентилі (специфічні для обладнання), вимірювання точності вентилів, глибини схеми та щільності вентилів (наскільки ефективно вентилі упаковані в схему) є критично важливим для розуміння продуктивності.

3. Бенчмаркінг на рівні схем (Circuit-Level Benchmarking) тестує повноцінні квантові схеми, щоб перевірити, наскільки добре вони працюють у реальних умовах. Бенчмарки в цій категорії використовують деякі з наведених нижче метрик для оцінки поведінки схем у міру їхнього масштабування за глибиною та складністю:

- Квантовий об'єм (Quantum Volume, QV): відображає здатність системи надійно виконувати дедалі складніші схеми. Ця метрика враховує

комбінацію кількості кубітів, точності вентилів, зв'язності (connectivity) та продуктивності компілятора.

- Кількість операцій на рівні схеми за секунду (Circuit Layer Operations Per Second, CLOPS): вимірює швидкість виконання квантовим процесором шарів схем QV. Це метрика продуктивності, орієнтована на пропускну здатність.
- Бенчмаркінг перехресної ентропії (Cross-entropy benchmarking, XEB): масштабована схема характеризації шуму та бенчмаркінгу, яка оцінює точність *n*-кубітної схеми, що складається з одно- та двокубітних вентилів.

4. Бенчмаркінг на рівні процесора (Processor-Level Benchmarking): Цей рівень оцінює загальну продуктивність квантових процесорів. Він включає загальносистемні метрики, такі як квантовий об'єм, комплексне тестування робочих навантажень (holistic workload testing) та час до отримання рішення (time-to-solution), який вимірює, наскільки швидко й точно пристрій може виконати завдання.

5. Бенчмаркінг на рівні платформи (Platform Level Benchmarking): зосереджується на ширшій квантовій екосистемі — хмарному доступі, компіляторах, SDK та API. Різні провайдери пропонують унікальні середовища із власними програмними стеками, такими як Qiskit від IBM, Cirq від Google, TKET від Quantinuum, Berkeley Quantum Synthesis Toolkit (BQSKit), Quantum Tool Suite (QTS) та Staq від Університету Дьюка. Усі вони пропонують різні ступені абстракції, оптимізації продуктивності та апаратної інтеграції [33]. Бенчмаркінг на цьому рівні допомагає порівнювати платформи за допомогою таких інструментів, як Open Quantum Assembly Language (QASMBench) або MQT Bench, які забезпечують узгоджене тестування різних апаратних технологій (наприклад, надпровідних, на основі захоплених іонів або фотонних кубітів).

6. Бенчмаркінг на рівні застосунків (Application-Level Benchmarking): стосується тестування того, наскільки добре квантові системи справляються з реальними проблемами. Такі фреймворки, як Quantum computing Application benchmark (QUARK), Munich Quantum Toolkit (MQT), та тести, орієнтовані на конкретні навантаження, перевіряють продуктивність з погляду застосунку. Вони

розглядають, як поводяться такі алгоритми, як варіаційна квантова факторизація (Variational Quantum Factoring, VQF), на різних системах.

7. Бенчмаркінг повного стека (Full Stack Benchmarking): пов'язує все разом, оцінюючи всю систему — від обладнання та схем до оркестрації, управління ресурсами та затримок. Це особливо корисно в застосунках ери NISQ, що включають гібридні квантово-класичні системи, де продуктивність на кожному рівні впливає на кінцевий результат.

Бенчмаркінг квантових систем вимагає низки метрик продуктивності для оцінки їхньої надійності, швидкості та точності обчислень. До найуживаніших метрик у фреймворках квантового бенчмаркінгу належать:

- Когерентність кубітів (Qubit coherence): Когерентність кубітів — це тривалість часу, протягом якого кубіт може зберігати свій квантовий стан до того, як відбудеться декогеренція. Когерентність має важливе значення для збереження суперпозиції та заплутаності. Час когерентності є критичним показником продуктивності, оскільки збурення навколишнього середовища можуть спричинити декогеренцію, що призводить до втрати квантової інформації та робить обчислення ненадійними [34].
- Середня точність вентиля (Average gate fidelity): Ця метрика кількісно визначає різницю між ідеальною операцією унітарного вентиля та її реальною реалізацією. Висока середня точність вентиля вказує на те, що квантовий вентиль поводиться подібно до свого теоретичного аналога, мінімізуючи операційні помилки [35].
- Точність схеми (Circuit fidelity): Точність схеми вимірює, наскільки безпомилково вся квантова схема виконує своє цільове завдання. Вона оцінює подібність між кінцевим станом фізичної системи та ідеальним (безшумовим) станом, відображаючи загальну коректність схеми та її стійкість до шуму [36].
- Рівень помилок (Error rate): У квантових обчисленнях рівень помилок позначає ймовірність того, що кубіт змінить свій стан під час обчислення. Сучасні квантові комп'ютери зазвичай мають рівень помилок у діапазоні від

0,1% до 1%, що значно вище за рівень помилок у класичних комп'ютерних системах [37].

- Квантовий об'єм (Quantum Volume): це метрика, що представляє найбільшу квадратну квантову схему (де ширина, або кількість кубітів, дорівнює глибині, або кількості шарів вентилів), яку процесор може успішно реалізувати. Вона враховує взаємодію кількості кубітів, точності вентилів, зв'язності та ефективності компілятора.
- Точність зчитування кубітів (Qubit readout fidelity): Також відома як точність вимірювання кубітів. Вона вказує на рівень помилок, пов'язаний з операціями зчитування кубітів. Ця метрика відображає ймовірність того, що виміряний результат справді відповідає фактичному стану кубіта на момент вимірювання.
- Час виконання квантового вентиля (Quantum gate execution time): Також називається швидкістю квантового вентиля. Цей показник вказує на час, необхідний для виконання операцій квантового вентиля. Більша швидкість виконання зазвичай дозволяє реалізувати глибші схеми в межах обмежених вікон когерентності.
- Точність SPAM (SPAM fidelity): Точність процесу підготовки стану та вимірювання (State Preparation and Measurement Process, SPAM) вказує на те, наскільки добре квантовий комп'ютер може ініціалізувати кубіти в потрібний стан і точно зчитати їхній стан після обчислень. Висока точність SPAM є важливою для надійних квантових операцій вводу-виводу [38].

Фреймворки бенчмаркінгу допомагають оцінити, наскільки ефективно виконуються завдання в цьому гібридному ландшафті, визначаючи найефективніші конфігурації та виявляючи «вузькі місця» в розподілі ресурсів, затримках та оркестрації [8,9]. Ці розподілені архітектури поєднують квантові процесори (QPU) та класичні обчислювальні блоки (CPU/GPU), розгорнуті на периферійних пристроях та в централізованій хмарній інфраструктурі. Мета полягає в тому, щоб використати обчислювальні переваги квантових технологій, зберігаючи при цьому масштабованість, керованість та швидкість реагування, які

пропонують класичні системи. Виклики в такому середовищі включають квантовий шум та виправлення помилок, обмеження затримки та пропускної здатності, складність програмного стека, питання безпеки та конфіденційності в розподілених системах, а також відсутність зрілих фреймворків для гібридної оркестрації.

У цьому контексті бенчмаркінг оцінює як обчислювальну, так і комунікаційну продуктивність гібридних систем. З обчислювального боку ключові фактори включають ефективність трансляції (transpilation) квантової схеми, точність виконання та результативність постобробки. З комунікаційного боку критичною метрикою є затримка між периферійним та хмарним рівнями, особливо для застосунків, чутливих до часу. Завдяки інтеграції квантових можливостей у системи «периферія-хмара» мета полягає у зменшенні загальної затримки, підвищенні обчислювальної пропускної здатності та покращенні масштабованості, зокрема для вимогливих завдань у таких сферах, як оптимізація, криптографія та штучний інтелект.

Для комплексного вимірювання продуктивності в цій статті пропонується використання показника затримки квантової системи «периферія-хмара» (quantum-edge-cloud latency score) — уніфікованої метрики бенчмаркінгу, яка відображає загальний час, витрачений на виконання обчислювального завдання, ініційованого периферійним пристроєм. Коли запит на обчислення надходить із домену IoT до системи «периферія-хмара», система виконує кілька ключових операцій: по-перше, вона транслює квантову схему відповідно до топології конкретного обладнання; далі вона виконує схему на квантовому процесорі або симуляторі; і, нарешті, повертає результат на шлюз IoT [39]. Загальний час, витрачений на ці етапи, включно з передачею запиту, обробкою, виконанням та відповіддю, враховується в показнику затримки.

Цей показник затримки (в одиницях часу) забезпечує наскрізну (end-to-end) оцінку швидкості реагування системи. Чим вищий показник затримки, тим повільніша та менш ефективна тестована система, тоді як нижчий показник затримки свідчить про ефективнішу обробку та/або комунікацію в континуумі

«периферія-хмара». Це робить цю метрику потужним індикатором здатності системи справлятися як з обчислювальними, так і з комунікаційними навантаженнями в динамічному середовищі з обмеженими ресурсами.

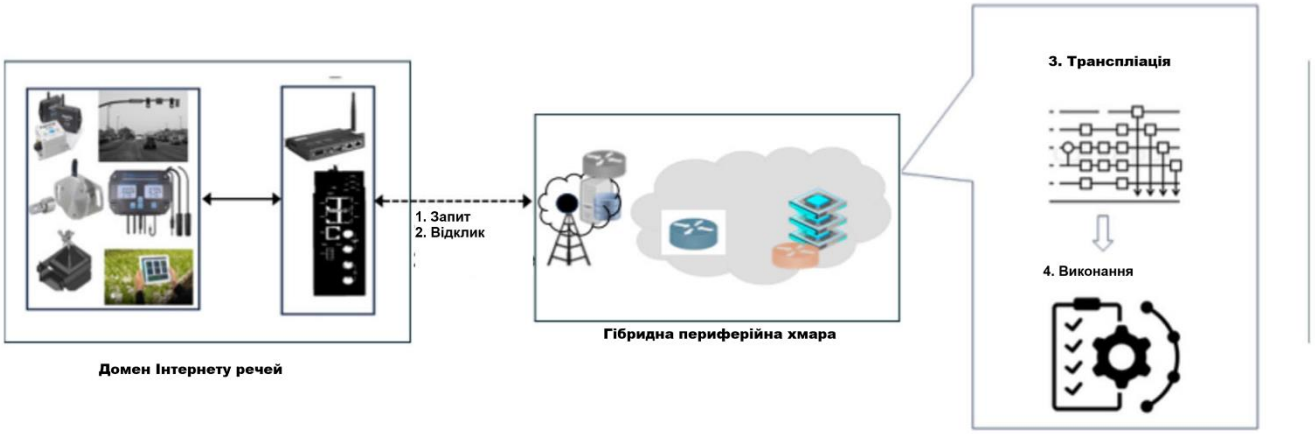

Рисунок 2.2 – Квантові обчислення в гібридному середовищі квантової крайової хмари.

На Рис. 2.2. проілюстровано квантовий робочий процес у гібридній квантово-класичній системі «периферія-хмара». Важливо включити етап трансляції до процесу бенчмаркінгу, щоб врахувати практичні обмеження реального квантового обладнання. Етапи 1 і 2 передбачають комунікацію між пристроєм та системою «периферія-хмара», тоді як етапи 3 і 4 охоплюють обробку схеми та її виконання на інфраструктурі «периферія-хмара». Загальний час проходження цих чотирьох етапів визначає показник затримки квантової системи «периферія-хмара» — ключову метрику для оцінки швидкості реагування системи.

Наскільки нам відомо, жоден з існуючих фреймворків бенчмаркінгу не розглядає інтеграцію квантових обчислень у середовища «периферія-хмара». Бенчмаркінг на системному рівні включає квантові та класичні обчислювальні, а також комунікаційні компоненти. Більшість доступних робіт зосереджені або на квантових обчисленнях, або на класичних мережах «периферія-хмара», не поєднуючи обидва домени — особливо з погляду врахування наскрізної затримки (end-to-end latency).

У квантовому домені існують проблемно-орієнтований (application-oriented) бенчмаркінг, бенчмаркінг на рівні оркестрації та архітектурні аспекти інтегрованих мереж «периферія-хмара». У цьому дослідженні розглядаються деякі відповідні наукові роботи, опубліковані після 2020 року, щоб виявити прогалини в дослідженнях та обґрунтувати необхідність цієї роботи. Дослідження в [40] вивчає фреймворки проблемно-орієнтованого бенчмаркінгу для оцінки таких метрик, як заплутаність та глибина шляху (path depth). Автори адаптують принципи класичного бенчмаркінгу до квантового домену, пропонуючи розрахунки балів (score calculations) та вектори ознак для кількісної оцінки охоплення застосунків.

Автори роботи [41] пропонують проблемно-орієнтований бенчмарк на основі квантових технологій, який вивчає вплив обмежень глибини квантових схем на реальні застосунки. Фінзгар та ін. пропонують фреймворк, який зосереджується спеціально на оцінці гібридних квантово-класичних алгоритмів з використанням предметно-орієнтованих застосунків, таких як логістика та планування в робототехніці [42]. Ці бенчмарки припускають доступність квантових обчислень лише в головній хмарі та не враховують затримку в системі «периферія-хмара». Аспекти оркестрації гібридних квантово-класичних систем вивчалися, зокрема, у контексті варіаційних алгоритмів. Робота в [43] демонструє покращення часу виконання та пропускної здатності завдяки оновленню параметрів на боці сервера та механізмам скидання кубітів. У [44] досліджується вплив зменшення накладних витрат на квантово-класичну комунікацію для прискорення виконання гібридних алгоритмів [45]. У [46] автори досліджують інтегровані квантово-класичні системи та підкреслюють необхідність інструментів оцінки продуктивності на системному рівні.

У [47] автори вводять метрику на основі корисності (utility-based metric) для оцінки операційної вигоди від розподілу квантових вузлів між периферійною та головною хмарами з урахуванням затримки, рівня успішності та точності (fidelity). Як було зазначено вище, зусилля з бенчмаркінгу інтегрованих квантово-класичних систем «периферія-хмара» є вкрай обмеженими. Для гібридних мереж квантових периферійних обчислень автори в [48] визначають затримку, синхронізацію між

периферією та хмарою, а також гетерогенність пристроїв як головні виклики. Робота [32] представляє MQT Bench — універсальну платформу бенчмаркінгу, яка підтримує кілька рівнів абстракції з широким набором бенчмарків та можливістю узагальнення на різних рівнях.

Дослідники в [41] вводять QASMBench — наскрізний набір, що містить різноманітні типи схем та інструменти оцінки як для попереднього, так і для посттрансляційного (post-transpilation) аналізу, охоплюючи дрібномасштабні, середньомасштабні та великомасштабні квантові системи. У роботі [33] представлено комплексний набір для бенчмаркінгу, який оцінює продуктивність різних основних наборів засобів розробки програмного забезпечення (SDK) для квантових обчислень. Різні типи квантових платформ також виступають як категорія бенчмарків. Наприклад, роботи [41,42] зосереджені на таких платформах, як IBM-Q, Rigetti, IonQ, Quantinuum та AQT@LBN.

Оскільки інтеграція квантових обчислень та середовища «периферія-хмара» все ще перебуває на початковій стадії, більшість існуючих досліджень зосереджені на перевагах у сфері конфіденційності, а не на питаннях продуктивності, таких як наскрізна затримка, що включає розподілені обчислення та взаємодію в мережі між периферійними та хмарними доменами. Існуючі фреймворки бенчмаркінгу, такі як SupermarQ та Quark, оцінюють централізовані гібридні алгоритми і не враховують квантові та класичні ресурси, розподілені на периферії та в головних хмарах. Багатодоменні архітектури залишаються поза увагою. Передбачається, що в умовах реального світу класичні та квантові пристрої будуть взаємодіяти один з одним як у межах одного, так і між різними доменами.

Трансляції (транспіляції) та її рівням оптимізації топології не приділяється увага як важливим факторам. Якщо квантовому комп'ютеру потрібно виконати високорівневі схеми, ці схеми мають бути перетворені (тобто трансльовані) у відповідні примітивні квантові вентилі конкретних квантових процесорів (QPU) [49]. На відміну від традиційної компіляції, яка перетворює вихідний код на цільову мову [50], трансляція (transpilation) цілеспрямовано перетворює квантову схему так, щоб вона відповідала топології та обмеженням заданого квантового

пристрою [50]. Наприклад, тоді як деякі компілятори забезпечують кілька рівнів оптимізації для класичних систем, багато квантових платформ також пропонують різні рівні оптимізації під час трансляції. Цей етап є ключовим, оскільки ефективне виконання квантових схем залежить від того, наскільки добре процес трансляції адаптує логічні з'єднання кубітів до фізичної архітектури QPU. Недостатня оцінка розподілу гетерогенних ресурсів. Бенчмарки не оцінюють, наскільки добре системи справляються з динамічним перенесенням обчислень з урахуванням наявних ресурсів або з адаптивним балансуванням навантаження в гібридних сценаріях «периферія-хмара».

2.4. Опис системи бенчмаркінгу

Розглянемо структуру запропонованої системи бенчмаркінгу. Ґрунтуючись на концептуальному алгоритмі виконання завдань, показаному на Рис. 2.2. для гібридного класико-квантового середовища IoT–Edgecloud, загальна затримка розкладається на три основні компоненти: затримка зв'язку, затримка транспіляції та затримка виконання, як показано на Рис.2.3.

Щоб оцінити вплив різних стратегій оптимізації, пропонуються два методи: Метод A, який використовує попередньо визначені порядкові рівні оптимізації (наприклад, рівні 0–3, як визначено платформою IBM), та Метод B, який використовує неперервні десяткові значення для більш точного представлення рівнів оптимізації. Це дослідження визначає показник затримки квантової системи «периферія-хмара» (позначається як L) як комплексну метрику для оцінки наскрізної затримки в обидва кінці у гібридній квантово-класичній системі «периферія-хмара». Цей показник охоплює всі ключові етапи конвеєра обробки даних: передачу запиту від пристрою IoT або периферійного пристрою, трансляцію квантової схеми, виконання квантової схеми та передачу результату назад до пристрою-ініціатора.

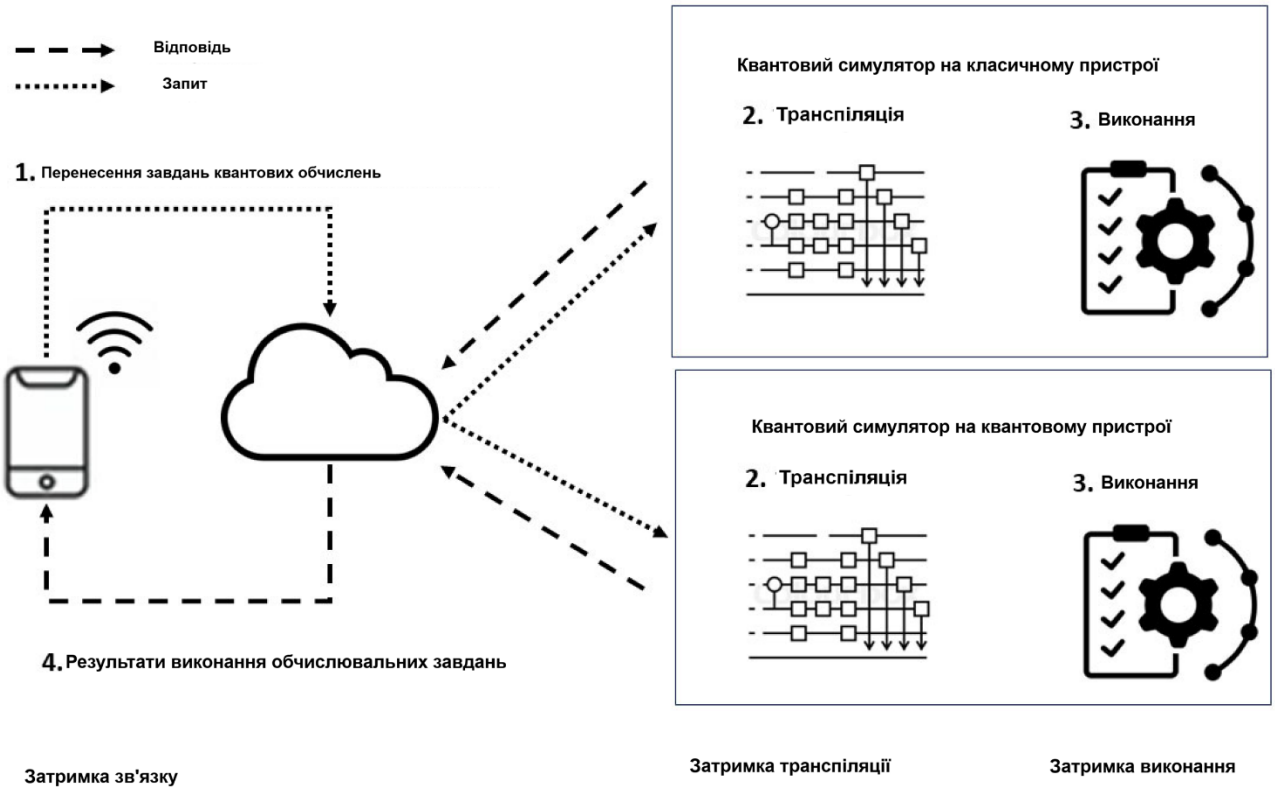

Рисунок 2.3 – Затримки під час обміну даними, транспіляції та виконання у гібридних робочих процесах.

Формально загальна затримка, або показник L (L-score), обчислюється як сума всіх затримок від пристроїв IoT до головної хмари. Його можна представити у такому вигляді:

$$L = \sum_{i=1}^{n} \tau_i \quad (3)$$

Ця робота обмежується трьома основними джерелами затримки, які суттєво впливають на продуктивність гібридних квантово-класичних систем «периферія-хмара». Ці компоненти пояснено нижче: • τ_{cl}: Комунікаційна затримка (Communication Latency) Вона означає час, необхідний для передачі даних між рівнем операційних технологій (OT), який складається з таких пристроїв, як датчики та пристрої IoT, та серверами системи «периферія-хмара». Вона включає в себе низку затримок, зумовлених мережевою передачею, маршрутизацією повідомлень, обробкою протоколів, перевантаженням мережі та апаратними

обмеженнями. Ці фактори в сукупності впливають на швидкість реагування та пропускну здатність системи.

- τ_{tl}: Затримка трансляції (Transpilation Latency) Це час, необхідний для перетворення високорівневої квантової схеми у форму, сумісну з фізичними обмеженнями конкретного процесора. Цей процес включає застосування попередньо визначених налаштувань оптимізації трансляції. На цьому етапі застосовуються попередньо визначені параметри оптимізації трансляції. Подібно до класичної компіляції, вищі рівні оптимізації трансляції призводять до збільшення часу трансляції.
- τ_{el}: Затримка виконання (Execution Latency) Цей показник вимірює час, витрачений сервером «периферія-хмара» на виконання трансльованої квантової схеми на QPU або на квантовому симуляторі. Як показано на етапі 4 на рисунку 4, ця затримка залежить від складності схеми, якості трансляції та обчислювальної продуктивності використовуваного квантового обладнання. У нашому випадку загальну затримку, або показник L, можна змоделювати як суму трьох описаних компонентів:

$$L = \tau_{cl} + \tau_{tl} + \tau_{el}. \tag{2.3}$$

Уніфікований показник L забезпечує комплексний та надійний критерій (бенчмарк) для оцінки наскрізної продуктивності гібридних квантово-класичних систем у середовищах «периферія-хмара». Ця метрика є особливо корисною для застосунків, де критично важливою є робота з низькою затримкою, таких як аналітика в реальному часі, автономні системи та квантове машинне навчання на периферії (edge-based quantum machine learning). Ретельна оцінка показника затримки дозволяє дослідникам і розробникам виявляти «вузькі місця» в продуктивності, оптимізувати конфігурації систем і покращувати інтеграцію квантових технологій в інфраструктуру «периферія-хмара».

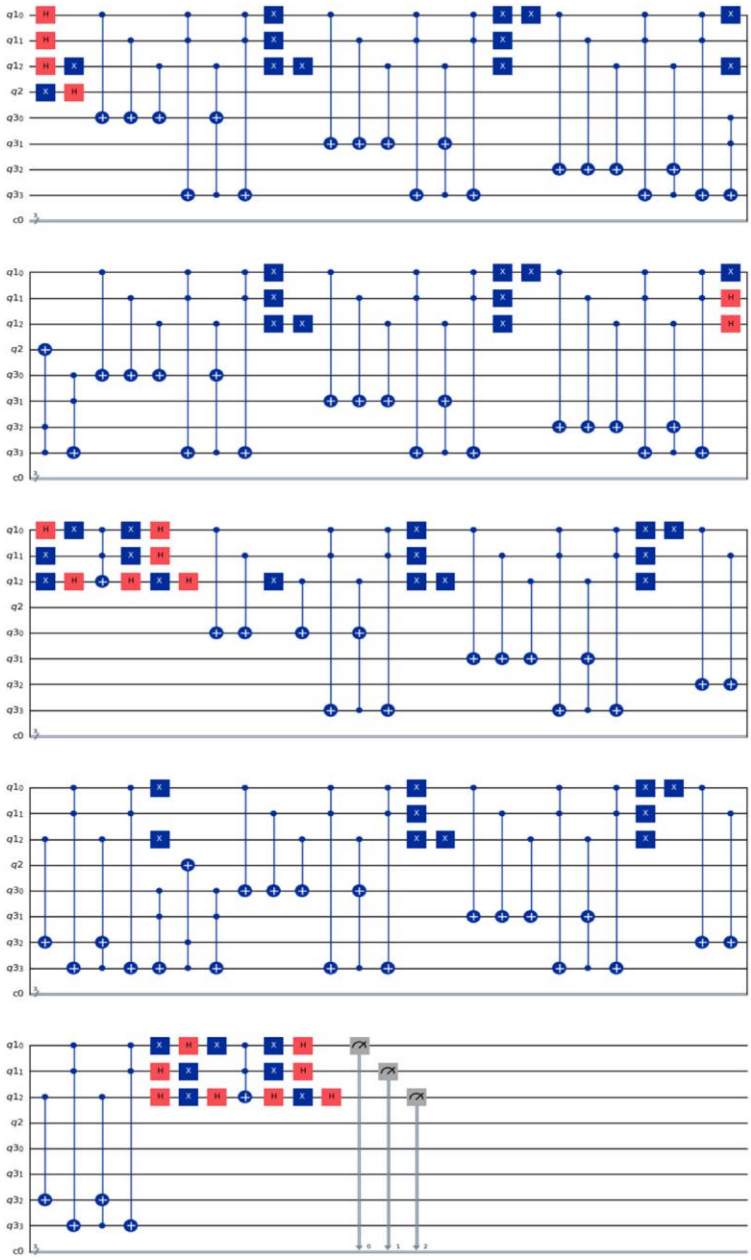

Рисунок 2.4 – Реалізація квантової схеми для алгоритму Шора в бібліотеці IBM Qiskit. На цьому рисунку $a = 7$ та $N = 15$, а сама схема потребує загалом 8 кубітів. Вона побудована за допомогою бібліотеки IBM Qiskit. Синій блок із позначкою X — це вентиль X (X-gate), який використовується для інверсії квантового стану. Червоний блок із позначкою H — це вентиль Адамара (Hadamard gate). Графічний елемент у вигляді синіх точок, з'єднаних лінією зі знаком «+», — це керований вентиль X (controlled X gate).

2.5. Чинники, які впливають показник затримки в квантових периферійних хмарних середовищах. Запропонована система бенчмаркінгу. Затримка зв'язку. Затримка транспіляції

Рівні оптимізації трансляції (транспіляції) та точність (fidelity) є керованими змінними, тоді як глибина схем і кількість кубітів насамперед визначаються специфікою реалізації квантових алгоритмів. Ключові фактори впливу на затримку:

Рівні оптимізації трансляції (Transpilation Optimization Levels), які забезпечуються платформами квантових обчислень, керують тим, як саме перетворюються схеми для кращої відповідності апаратним обмеженням. Мета полягає в тому, щоб визначити найбільш ефективну конфігурацію, яка балансує між часом оптимізації та обчислювальною продуктивністю. Наприклад, на платформі IBM: o Рівень 0 означає відсутність оптимізації; o Рівень 1 об’єднує однокубітні вентилі та скасовує послідовні вентилі CNOT; o Рівень 2 видаляє надлишкові операції; o Рівень 3 застосовує агресивні оптимізації, включаючи відображення кубітів із урахуванням шуму (noise-aware qubit mapping) та вдосконалене скасування вентилів [51,52].

Точність (Fidelity). Він є мірою того, наскільки безпомилково функціонує квантова система, особливо коли шум впливає на надійність і швидкість виконання [53–55]. Точність між двома квантовими станами, представленими матрицями густини ρ та σ, обчислюється як:

$$F(\rho,\sigma) = tr\left(\sqrt{\sqrt{\sigma}\rho\sqrt{\sigma}}\right). \tag{2.4}$$

У цьому формулюванні обидва стани можуть бути змішаними квантовими станами, які є статистичними ансамблями чистих станів [56]. Коли два стани майже ідентичні, точність наближається до 1, що свідчить про високу точність обчислень.

Глибина схеми (Circuit depth): У квантових обчисленнях це кількість послідовних шарів квантових вентилів, необхідних для реалізації квантового алгоритму [49]. Глибші схеми потребують більше часу на виконання, що збільшує

ймовірність помилок через декогеренцію кубітів та недосконаліть вентилів у пристроях NISQ. Крім того, глибина схеми впливає як на процес трансляції, так і на вразливість до шуму. Методи оптимізації під час трансляції спрямовані на зменшення глибини схеми при збереженні алгоритмічної цілісності.

Кількість кубітів: Кількість кубітів у квантовому алгоритмі є фундаментальним фактором, що визначає продуктивність системи, оскільки вона безпосередньо впливає на обчислювальну потужність, рівень помилок і потреби в ресурсах [57]. Зі збільшенням кількості кубітів квантова система може представляти експоненціально більші простори станів, що дозволяє розв'язувати складніші задачі. Однак таке масштабування пов'язане зі значними викликами, серед яких чутливість до шуму [58], підвищений рівень помилок вентилів, а також триваліший час трансляції та виконання.

Метрика комунікаційної затримки є критично важливим компонентом загальної продуктивності системи, особливо в застосунках, чутливих до затримок, таких як аналітика в реальному часі та квантове виведення (інференс) на периферії (edge-based quantum inference). Вона відображає загальний круговий час (round trip time, RTT), необхідний для передачі даних від пристроїв операційних технологій (OT), таких як датчики IoT, до серверів системи «периферія-хмара», час обробки та формування відповіді на сервері, а також час, необхідний для повернення відповіді до пристроїв-ініціаторів. Використовувані процесори та канали зв'язку можуть бути як класичними, так і квантовими. Формула для цього розрахунку виглядає так:

$$\tau_{cl} = \tau_{request} + \tau_{setup} + \tau_{response}, \tag{2.5}$$

де всі показники вимірюються в секундах, $\tau_{request}$ означає час передачі від пристрою OT до сервера «периферія-хмара»:

$$\tau_{request} = \frac{|\,\text{requestmessage}\,|}{\text{NetworkSpeed}}, \tag{2.6}$$

τ_{setup} означає час обробки комунікації на сервері (наприклад, автентифікація, встановлення з'єднання, обробка повідомлень тощо), $\tau_{response}$ означає час, необхідний для отримання відповіді назад на пристрої OT:

$$\tau_{call} = \tau_{response} = \frac{|\text{responsemessage}|}{\text{NetworkSpeed}}. \quad (2.7)$$

Ця метрика дозволяє точно оцінювати затримки, пов'язані з мережею, і допомагає виявляти можливості для оптимізації в інфраструктурі «периферія-хмара».

Затримка транспіляції — Метод A Затримка транспіляції (τ_{tl}^{A}) вимірює час, витрачений на процедуру транспіляції. Вона може бути виражена так:

$$\tau_{tl}^{A} = \frac{\sum_{i=1}^{N} \tau_{transpilation}^{i} \cdot i}{N}. \quad (2.8)$$

Тут N — це загальна кількість рівнів оптимізації для транспіляції на конкретній платформі, які надаються квантовими API. Наприклад, квантова платформа IBM пропонує 4 різних рівні оптимізації; таким чином, $N = 4$ для платформи IBM. Член $\tau_{transpilation}^{i}$ позначає затримку транспіляції для конкретного рівня оптимізації i. Параметр i є порядковим значенням i-го рівня оптимізації для транспіляції. Хоча на платформі IBM рівні оптимізації починаються з 0, у всіх формулах це значення зазвичай ініціалізується з 1.

Затримка транспіляції — Метод B Затримка транспіляції (τ_{tl}^{B}) також використовується для вимірювання часу, витраченого на процедуру транспіляції, але з іншими ваговими коефіцієнтами:

$$\tau_{tl}^{B} = \frac{\sum_{j=1}^{N} \tau_{transpilation}^{j} \cdot \text{weight}_{j}}{N}. \quad (2.9)$$

Тут N — це загальна кількість рівнів оптимізації, що відповідає рівнянню (2.9). Член $\tau_{transpilation}^{j}$ позначає затримку транспіляції на рівні оптимізації j. Змінна j відіграє роль, аналогічну i в рівнянні (2.9), але, на відміну від i, її значення також відображає її величину. Наприклад, у рівнянні (2.9) встановлення $i = 3$ неявно передбачає вагу, яка в 1,5 раза перевищує вагу для $i = 2$ для всіх алгоритмів. Однак експериментальні результати, представлені в розділах 5 і 6, не підтверджують це

припущення про рівномірне масштабування. Щоб вирішити цю проблему, зважені рівні оптимізації j адаптуються для різних квантових алгоритмів, забезпечуючи збереження властивостей точності, встановлених у рівнянні (2.9), також і в рівнянні (2.10). Співвідношення зважування виражається так:

$$\text{weight}_j = \frac{\sum_{k=1}^{M} \frac{\tau_{transpilation}^{j,k}}{\tau_{transpilation}^{1,k}}}{M}. \quad (2.10)$$

Тут j використовується для вираження рівня оптимізації транспіляції, який може набувати десяткових значень. Ваги усереднюються за всіма розглянутими квантовими алгоритмами. M — це загальна кількість використовуваних алгоритмів, а k є індексом окремих алгоритмів. У цьому дослідженні $k \in \{1,2,3\}$, оскільки обрано 3 квантові алгоритми. Член $\tau_{transpilation}^{j,k}$ — це час транспіляції алгоритму k на рівні транспіляції j, тоді як $\tau_{transpilation}^{1,k}$ — це час операції транспіляції для рівня без оптимізації (нульового рівня).

Затримка виконання — Метод А Рівняння (2.10) визначає затримку виконання на сервері «периферія-хмара». У ньому використовуються порядкові значення i як рівні транспіляції, відтак $\tau_{execution}^{i}$ позначає час виконання для рівня транспіляції i.

$$\tau_{execution}^{A} = \frac{\sum_{i=1}^{N} \frac{\tau_{execution}^{i}}{\text{Fidelity} \cdot i}}{N}. \quad (2.11)$$

Затримка виконання — Метод В Рівняння (13) представляє затримку виконання на сервері «периферія-хмара», враховуючи ваговий коефіцієнт weight_j, наведений у рівнянні (14). Обидва рівняння (12) і (13) відображають припущення, що вища точність (Fidelity) призводить до меншої затримки обчислень і, як наслідок, до підвищення продуктивності.

$$\tau_{execution}^{B} = \frac{\sum_{j=1}^{N} \frac{\tau_{execution}^{j}}{\text{Fidelity} \cdot \text{weight}_j}}{N}, \quad (2.12)$$

$$\text{weight}_j = \frac{\sum_{k=1}^{M} \frac{\tau_{execution}^{j,k}}{\tau_{execution}^{1,k}}}{M}. \quad (2.13)$$

Тут weight_j виражає відносну вагу виконання, що відповідає рівню оптимізації транспіляції j, представленому в десяткових значеннях (часові показники, зумовлені рівнем оптимізації транспіляції у вигляді десяткових чисел). Визначення M та k відповідають тим самим дефініціям, що й у рівнянні (2.10). Значення $\tau_{execution}^{j,k}$ позначає час виконання алгоритму k на рівні транспіляції j, тоді як $\tau_{execution}^{1,k}$ відповідає часу виконання алгоритму k без оптимізації.

3. ЕКСПЕРИМЕНТАЛЬНА РЕАЛІЗАЦІЯ КВАНТОВИХ АЛГОРИТМІВ, ОЦІНКА ЕФЕКТИВНОСТІ ПРОГРАМНОЇ СИСТЕМИ

Підхід розвинений у дипломній роботі використовує два незалежні середовища виконання для запуску квантових схем: фізичні квантові процесори (QPU) та квантові симулятори. Симулятори — це програмне забезпечення, яке моделює роботу QPU на класичних комп'ютерах, дозволяючи тестувати та валідувати алгоритми без необхідності доступу до реального квантового обладнання.

3.1. Квантові алгоритми, обрані для оцінки затримки.

У цьому дослідженні для проведення експериментів та оцінки продуктивності використовуються дві провідні платформи квантових обчислень: IBM Quantum Lab (Йорктаун-Гайтс, штат Нью-Йорк, США) та Amazon Braket (Сіетл, штат Вашингтон, США). Обидві платформи надають доступ як до надпровідних QPU, так і до симуляторів. Характеристики дослідницьких платформ такі:

- IBM Quantum Lab надає хмарний доступ до власних квантових процесорів IBM, включаючи QPU з кількістю надпровідних кубітів до 1000 [14,58]. Ці QPU розроблені для виконання високоточних квантових операцій, що робить платформу надзвичайно придатною для виконання складних квантових алгоритмів. Метрики продуктивності IBM використовуються для оцінки надійності апаратного забезпечення та обчислювальних можливостей. Платформа підтримується Qiskit (v1.2.0) — набором засобів розробки програмного забезпечення (SDK) з відкритим вихідним кодом, який дозволяє дослідникам проєктувати, симулювати та виконувати квантові алгоритми як на реальних QPU, так і на симуляторах [15].
- Amazon Braket, що надається Amazon Web Services (AWS), пропонує уніфікований інтерфейс до низки QPU та симуляторів від кількох

постачальників апаратного забезпечення [59]. Він підтримує розробку та розгортання квантових алгоритмів за допомогою Python (v3.10), дозволяючи користувачам реалізовувати як суто квантові, так і гібридні квантово-класичні робочі процеси. Його модульна архітектура чудово підходить для дослідження передових застосунків в оптимізації, машинному навчанні та криптографії. Емуляція середовища «периферія-хмара» Завдяки використанню обох платформ це дослідження пропонує комплексну оцінку їхніх можливостей, висвітлюючи їхні відповідні сильні сторони та придатність для різних завдань квантових обчислень. Зазначається також, що для змістовної роботи з усіма типами затримок більшість експериментів передбачають перенесення обчислень (офлоудинг) до головної хмари.

Користувачі можуть створювати, тестувати та запускати на них квантові алгоритми, задіюючи як квантові, так і класичні обчислювальні ресурси. Виконання обраних алгоритмів на цих ресурсах залучає комунікаційну мережу, яка належить до телекомунікаційної інфраструктури, та канали зв’язку, що є власністю постачальника хмарних послуг. Це дозволяє емулювати наскрізні затримки в гібридному середовищі «периферія-хмара». Фактор периферійної обробки (edge processing) враховано через спрощене впровадження попередньої обробки на локальному сервері шляхом симуляції квантових процесорів на локальних CPU.

Для оцінки затримки в гібридних квантово-класичних середовищах «периферія-хмара» було обрано три репрезентативні квантові алгоритми: алгоритм Шора, алгоритм Гровера та алгоритм квантових блукань. Ці алгоритми представляють різні категорії парадигм квантових обчислень: квантове перетворення Фур’є, підсилення амплітуди та квантові блукання відповідно.

Алгоритм Шора обрано через його здатність виконувати факторизацію цілих чисел за поліноміальний час $O((\log N)^3)$, що є критично важливим для оцінки потужності квантових систем у зламі шифрування.

```
from qiskit import QuantumCircuit
from qiskit.circuit.library import QFT

#                                  : 3
                              + 2
shor_circuit = QuantumCircuit(5)

# 1.


shor_circuit.h([0, 1, 2])

#
                                                        |1>
shor_circuit.x(3)
shor_circuit.barrier()

# 2.
                     (
                      )
shor_circuit.cx(2, 3)
shor_circuit.cx(1, 4)
shor_circuit.barrier()

# 3.
                                                                ’    (
    IQFT)
iqft_block = QFT(num_qubits=3).inverse()
shor_circuit.append(iqft_block, [0, 1, 2])
shor_circuit.barrier()

#

fig = shor_circuit.draw(output=’mpl’)
fig.savefig(’shor_circuit.png’, dpi=300, bbox_inches=’tight’)
```

Алгоритм Гровера вибрано завдяки його ефективності під час пошуку в невідсортованих базах даних за рахунок зменшення складності неструктурованого пошуку з $O(N)$ до $O(\sqrt{N})$.

```
from qiskit import QuantumCircuit

#                                                    2
grover_circuit = QuantumCircuit(2)

# 1.                          :
grover_circuit.h([0, 1])
grover_circuit.barrier() #


# 2.                                  (
                                        |11>)
grover_circuit.cz(0, 1)
grover_circuit.barrier()

# 3.                                                (
                                                   )
grover_circuit.h([0, 1])
grover_circuit.x([0, 1])
grover_circuit.cz(0, 1)
grover_circuit.x([0, 1])
grover_circuit.h([0, 1])
grover_circuit.barrier()

#

fig = grover_circuit.draw(output='mpl')
fig.savefig('grover_circuit.png', dpi=300, bbox_inches='tight
    ')
```

Алгоритм квантових блукань включено для оцінки можливостей ймовірнісного моделювання та симуляції квантової системи, які є актуальними в таких прикладних сферах, як маршрутизація в мережах та прогнозне обслуговування (predictive maintenance). Ці алгоритми реалізовано на платформах IBM Quantum Lab та Amazon Braket для оцінки затримки за різних конфігурацій системи та стратегій трансляції (транспіляції).

```
from qiskit import QuantumCircuit

# 3                :              0                    ,
    1       2
qw_circuit = QuantumCircuit(3)

# 1.
                  (                                )
qw_circuit.h(0)
qw_circuit.barrier()

# 2.                                (Shift Operator)
#                                   |1>,
                                (+1)
qw_circuit.ccx(0, 1, 2)
qw_circuit.cx(0, 1)

#                                   |0>,
                              (-1)
qw_circuit.x(0)
qw_circuit.cx(0, 1)
qw_circuit.x(0)
qw_circuit.barrier()

#

fig = qw_circuit.draw(output='mpl')
fig.savefig('quantum_walk_circuit.png', dpi=300, bbox_inches=
    'tight')
```

Алгоритм Шора є ключовим квантовим алгоритмом, який ефективно розкладає на множники великі цілі числа, створюючи загрозу для класичних схем шифрування, що базуються на складності факторизації [10]. Центральна концепція алгоритму Шора полягає у визначенні періоду r специфічної функції модульної арифметики $f(x) = a^x \bmod N$, де a — це випадково обране ціле число, взаємно просте зі складеним числом N. Мета алгоритму — визначити найменше додатне ціле число r, таке що:

$$a^{x+r} \equiv a^x (\bmod N). \tag{3.1}$$

Після того як період r визначено за допомогою квантової підпрограми (квантового оцінювання фази), кроки класичної пост-обробки обчислюють найбільший спільний дільник (НСД) чисел $a^{r/2} \pm 1$ та N, що дає прості множники

числа N. Цей механізм пошуку періоду, що виконується на квантовому комп'ютері, демонструє чітку квантову перевагу у розв'язанні задачі факторизації цілих чисел за поліноміальний час $O(\log N)$, на відміну від класичних алгоритмів, яким для досить великих N знадобиться практично нездійсненна кількість часу. Таким чином, алгоритм Шора підкреслює трансформаційний потенціал квантових обчислень у таких галузях, як криптографія, оптимізація та безпечний зв'язок. На рисунку 4 показано реалізацію алгоритму Шора на платформі IBM. Окрім криптографії, алгоритм Шора також слугує критерієм (бенчмарком) для оцінки глибини виконання, точності (fidelity) та масштабованості систем у пристроях ери NISQ.

Алгоритм Гровера — це квантовий алгоритм пошуку, який забезпечує квадратичне прискорення порівняно з класичними підходами при пошуку в невідсортованій базі даних або розв'язанні певних задач із функцією «чорної скриньки» [11]. Тоді як класичним алгоритмам потрібно $O(N)$ запитів для знаходження цільового елемента в базі даних із N елементів, алгоритм Гровера скорочує цю кількість до $O(\sqrt{N})$, що робить його основоположним прикладом квантової алгоритмічної переваги. На Рис 3.1 представлено реалізацію алгоритму Гровера з використанням 3 кубітів для рядка «010», як це реалізовано в бібліотеці IBM Qiskit [16]. Алгоритм оперує функцією $f(x)$, такою що $f(x)=1$, якщо x є правильним розв'язком $\hat{x}$, і $f(x)=0$ в іншому випадку. Метод Гровера починається зі створення квантової суперпозиції всіх можливих розв'язків. Потім він використовує квантовий оракул для інверсії амплітуди правильного розв'язку $\hat{x}$. Після цього процес підсилення амплітуди збільшує ймовірність вимірювання правильного розв'язку шляхом систематичного підсилення позначеного стану.

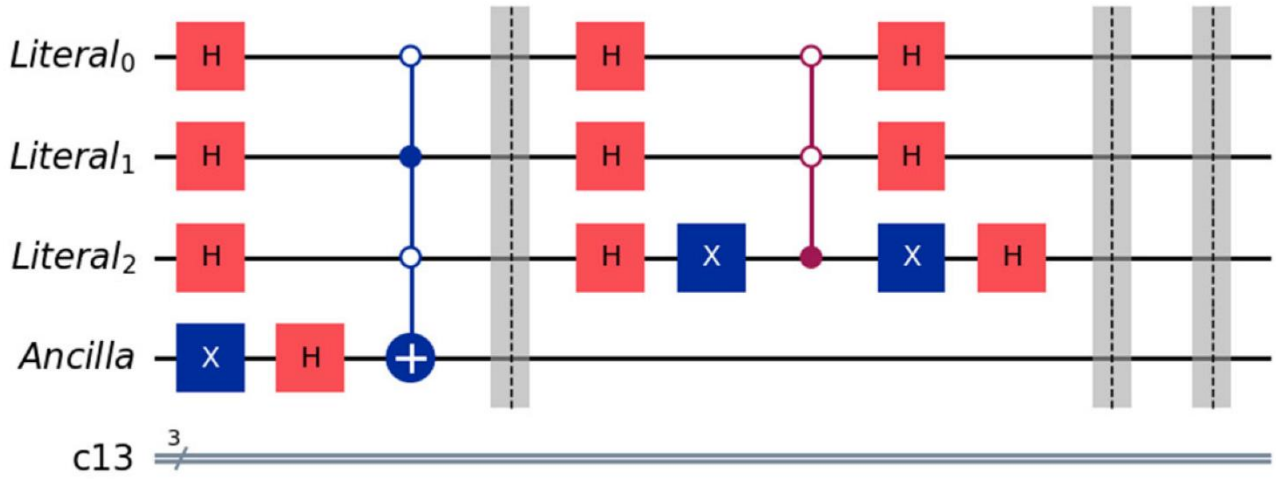

Рисунок 3.1 – Реалізація алгоритму Гровера з використанням 3 кубітів для рядка «010». Синій блок із позначкою X — це вентиль X (X-gate), червоний блок із позначкою H — це вентиль Адамара (Hadamard gate), а графічний елемент у вигляді синіх точок, з'єднаних лінією зі знаком «+», — це керований вентиль X (controlled X gate).

Відповідно до [60], ітераційна процедура керується оператором Гровера G, який визначається як:

$$G = (2|\psi\rangle\langle\psi| - I) \cdot (I - 2|\hat{x}\rangle\langle\hat{x}|). \tag{3.2}$$

Тут: $|\psi\rangle$ представляє стан рівномірної суперпозиції; символ I є одиничним оператором (оператором тотожності). Перший член в операторі здійснює відображення відносно середньої амплітуди, тоді як другий член застосовує інверсію фази до позначеного стану $\hat{x}$. Повторення цього оператора приблизно $O(\sqrt{N})$ разів підсилює амплітуду правильного стану, що дозволяє визначити його з високою ймовірністю під час вимірювання. Алгоритм Гровера є особливо цінним у таких галузях, як оптимізація [61,62], криптографія [11] та квантове машинне навчання [63], де неструктурований пошук відіграє центральну роль. На Рис. 3.1. наведено приклад алгоритму квантових блукань, реалізованого в [16]. Процедуру виконання алгоритму Гровера описано на Рис. 3.2.. На кроці 1 відбувається ініціалізація даних. 2. Крок 2 і крок 3 працюють разом у циклі для обробки функцій

підсилення (amplification) та оракула. 3. Крок 4 є фінальним кроком для повернення результату обчислень.

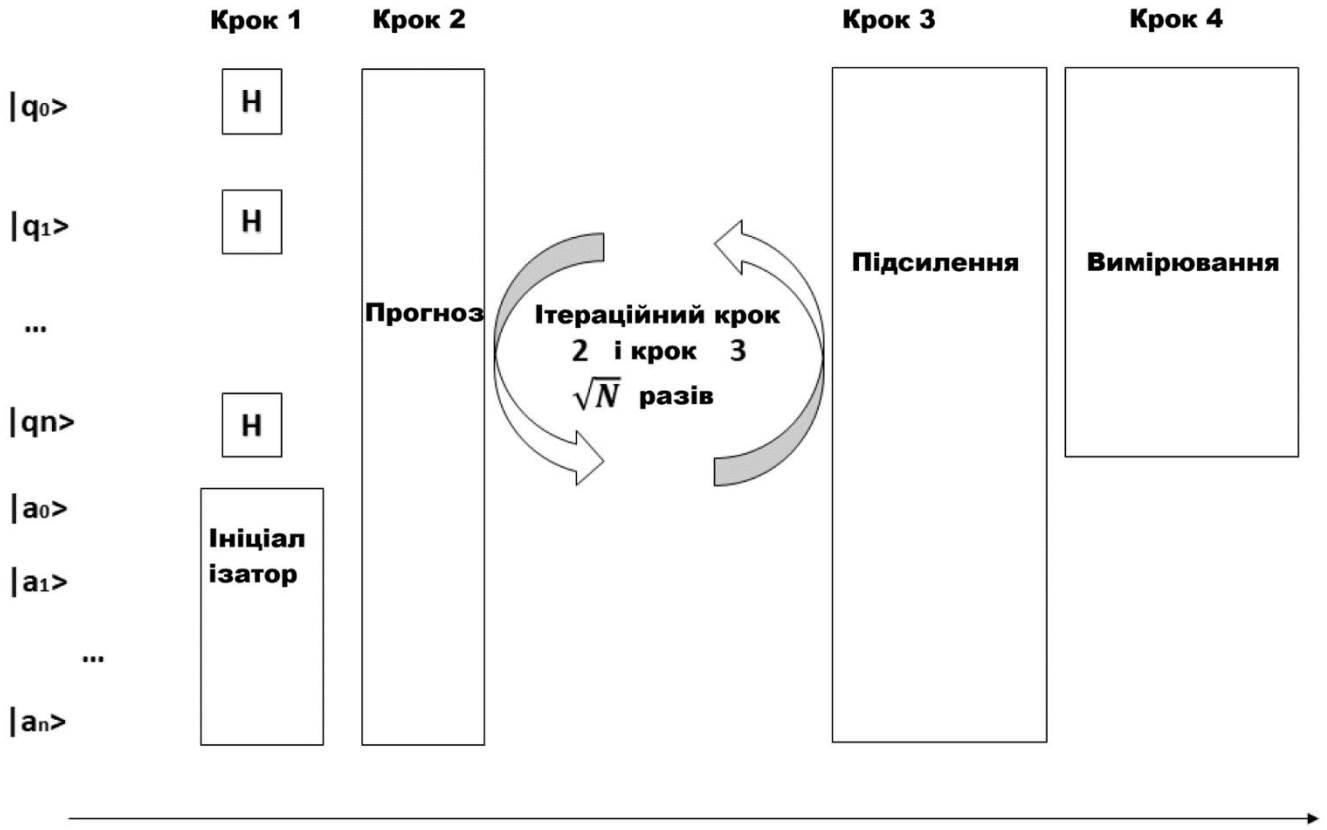

Рисунок 3.2 – Після кроку 1, який є ініціалізацією, крок 2 і крок 3 повторюються лише протягом $O(\sqrt{N})$ ітерацій.

Алгоритм квантових блукань є квантовою версією класичних випадкових блукань, які моделюють стохастичні процеси, де наступний стан системи ймовірнісно визначається її поточним станом [13]. Він забезпечує ефективніше дослідження складних і багатовимірних структур, що робить його чудово придатним для застосування в обході графів, аналізі мереж та оптимізації. На відміну від класичних випадкових блукань, квантові блукання використовують квантову суперпозицію та інтерференцію, уможливлюючи ефективніше дослідження складних структур. На Рис. 3.3 наведено приклад реалізації квантового блукання, який подано в бібліотеці IBM Qiskit [16]. Існує два типи квантових блукань:

- Квантові блукання з дискретним часом (DTQW)
- Квантові блукання з неперервним часом (CTQW).

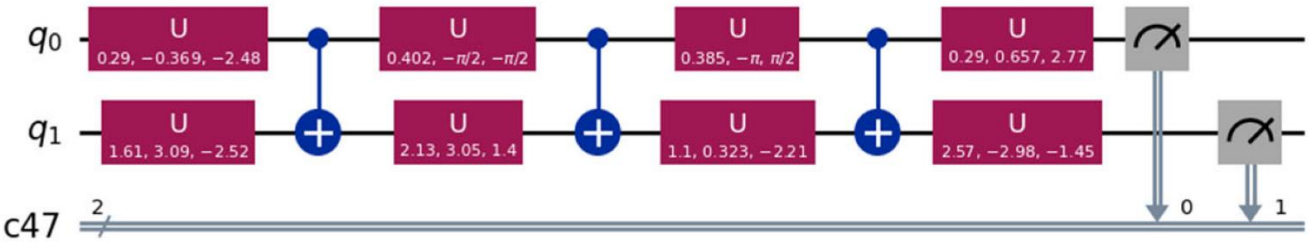

Рисунок 3.3 – Реалізація алгоритму квантового блукання з використанням 2 кубітів.

DTQW явно не містять параметра часу і зазвичай реалізуються через послідовність унітарних операцій. Натомість CTQW можна інтерпретувати як спеціалізовану форму квантової симуляції, що еволюціонує під дією незалежного від часу гамільтоніана, похідного від структури базового графа [64].

Натомість CTQW (квантові блукання з неперервним часом), які походять від марковських ланцюгів [64], керуються диференціальним рівнянням, що включає матрицю Лапласа (лапласіан) графа L, яка кодує зв'язність графа. Для графа G, вираженого через матрицю суміжності A, матриця Лапласа становить $L = D - A$, де $D = diag(\sum_j A_{1j}, \ldots, \sum_j A_{nj})$ — це матриця степенів вершин [63]. На відміну від DTQW, CTQW визначаються на гільбертовому просторі без необхідності додаткового простору монети [65]. Нехай $p_i(t)$ — ймовірність перебування у вершині i в момент часу t, тоді еволюція стану суперпозиції $|\Psi(t)\rangle$ для CTQW виражається як:

$$|\Psi(t)\rangle = e^{-iHt} |\Psi(0)\rangle, \tag{3.2}$$

де $H = \gamma L$.

Алгоритм квантових блукань продемонстрував значний потенціал у розв'язанні мережевих задач, таких як пошук найкоротших шляхів [66], виявлення спільнот [67] та аналіз зв'язності. Їх застосування виходить за межі теорії графів і

охоплює такі галузі, як оптимізація, машинне навчання та розпізнавання образів, де здатність ефективно обробляти великі та складні набори даних є критично важливою. Завдяки використанню властивих квантовим системам паралелізму та інтерференції, алгоритм квантових блукань є потужним інструментом для розв'язання обчислювально складних задач ефективніше, ніж це роблять їхні класичні аналоги.

3.2. Тестування програмної системи. Аналіз результатів

Комунікація є спільною як для Методу А, так і для Методу В, оскільки обидва передбачають передачу та прийом даних через ту саму конфігурацію мережі. Для моделювання реальних умов розгорнуто випробувальний стенд (testbed) на базі VirtualBox. Два віртуальні екземпляри синхронізовано за допомогою ідентичних системних годинників, а скрипти Python використовуються для емуляції обміну повідомленнями між периферійним пристроєм (edge device) та постачальником квантових послуг. Використовуючи базову пропускну здатність мережі 56 кбіт/с, час кругового зв'язку вимірюється на основі рівнянь (2.7) та (2.8). Хоча 56 кбіт/с обрано для простоти, цей параметр можна налаштувати для моделей у різних мережевих сценаріях.

Комунікаційна затримка: тематичне дослідження з алгоритмом Гровера Алгоритм Гровера можна адаптувати до широкого спектра задач пошуку. У нашому експерименті для практичної оцінки комунікаційної затримки алгоритм Гровера реалізовано з використанням оракула, призначеного для ідентифікації бінарного значення «110», представленого за допомогою 3 кубітів. Повідомлення запиту має таку структуру: name: 'grover', n: 110. Це повідомлення має розмір 27 байт і надсилається від периферійного пристрою до квантового сервера. На платформі IBM Quantum:

Таблиця 3.1. Повідомлення на платформі IBM для алгоритму Гровера.

Рівні оптимізації / Метрика	На симуляторі	На QPU
1	['110': 95, '000': 3, '101': 2]	['110': 39, '100': 14, '111': 13, '011': 8, '001': 6, '101': 6, '000': 8, '010': 6]
…	…	…
Довжина повідомлення (у байтах)	222	340

Таблиця 3.2. Повідомлення на платформі Amazon Braket для алгоритму Гровера.

Платформа	На симуляторі	На QPU
Amazon Braket	[0.03 0.04 0.06 0.02 0.03 0.03 0.76 0.03]	[0.02 0.1 0.09 0.08 0.06 0.02 0.53 0.1]
Довжина повідомлення (в байтах)	48	46

- Симулятори повертають вихідні дані у вигляді гістограми квантових станів та кількості їх появ.
- QPU повертають результати вимірювань у схожому форматі, хоча й з більшою варіативністю через шум.

  Наприклад:
- Вивід симулятора (Рівень оптимізації 1): ’110’: 95, ’000’: 3, ’101’: 2 Вказує на сильний пік для цільового стану.
- Вивід QPU (Рівень оптимізації 1): ’110’: 39, ’100’: 14, ’111’: 13, ’011’: 8, ’001’: 6, ’101’: 6, ’000’: 8, ’010’: 6 Відображає вплив шуму на результати вимірювань.
- На платформі Amazon Braket: Вихідні дані є розподілом ймовірностей для всіх 8 можливих 3-кубітних станів: [0.03, 0.04, 0.06, 0.02, 0.03, 0.03, 0.76, 0.03] Це відповідає станам [’000’, ’001’, ’010’, ’011’, ’100’, ’101’, ’110’, ’111’], причому стан 110 має найвищу ймовірність (0.76), як і очікувалося.

Загальний розмір повідомлень у відповідь, включаючи результати вимірювань та метадані, зафіксовано в Таблиці 3.1, і Таблиці 3.2, що дозволяє точно

розрахувати складову комунікаційної затримки. Це тематичне дослідження ілюструє варіативність форматів повідомлень та обсягів даних на різних платформах і в середовищах виконання, яку необхідно враховувати під час бенчмаркінгу затримки в гібридних квантово-класичних системах.

3.3. Аналіз затримки зв'язку в застсованих алгоритмах

В експериментальній реалізації алгоритму Шора параметри було встановлено як $a = 7$ та $N = 15$, що узгоджується з рівнянням (15). Ці значення налаштовують квантову схему на виконання факторизації цілих чисел шляхом визначення періоду r функції:

$$f(x) = a^x \bmod N. \tag{3.3}$$

Така конфігурація потребує загалом 8 кубітів для квантової схеми, щоб представити необхідний простір квантових станів. Повідомлення запиту для цього обчислення закодовано у форматі JSON як name: ’shors’, a: 7, N: 15. Це дає розмір повідомлення 39 байт. Після завершення обчислень система повертає результати вимірювань у форматах, специфічних для кожної платформи. У Таблиці 3.3 наведено відповідь від симулятора IBM на рівні оптимізації 1 із такими результатами вимірювань, як: [’10000000’: 27, ’0000000’: 27, ’01000000’: 21, ’11000000’: 25] Кожен 8-бітний бінарний рядок представляє квантовий стан, а пов’язане з ним значення вказує на кількість разів, коли цей стан спостерігався протягом кількох запусків. Наприклад, стан 10000000 з’являвся 27 разів, а 01000000 — 21 раз. У Таблиці 3.4 показано такі самі відформатовані повідомлення у відповідь, які повертає платформа Amazon.

Таблиця 3.3. Повідомлення у відповідь на платформі IBM для алгоритму Шора.

Рівні оптимізації	На симуляторі	На QPU
Рівень 1	['10000000': 27, '0000000': 27, '01000000': 21, '11000000': 25]	['10000100': 1, '01101111': 1, '10100010': 2, ..., '01001110': 2]
...	...	...
Загальна довжина повідомлення (у байтах)	264	558

Таблиця 3.4. Повідомлення у відповідь на платформі Amazon Braket для алгоритму Шора.

Платформа	На симуляторі	На QPU
Amazon Braket	['00001110': 10, '00000010': 8, ..., '11001011': 2]	['11111111': 20, '11110111': 7, ..., '11111010': 1]
Загальна довжина повідомлення (у байтах)	256	828

Далі алгоритм квантових блукань (QuantumWalks) реалізовано на круговому графі з 4 вузлами для дослідження його ймовірнісної та динамічної поведінки за різної кількості кроків. Аналіз зосереджений на кількості поява квантових станів після переходів в 1, 2, 3 та 4 кроки, що є критично важливим для розуміння ймовірнісної природи алгоритму.

Повідомлення запиту, що ініціює виконання, закодовано у форматі JSON як: name: ’quantumwalk’, n: 4, step: 4 Це дає загальний обсяг даних у 42 байти. В експерименті квантові блукання оцінюються на обох платформах: IBM Quantum та Amazon Braket. У Таблицях 3.5, 3.6, 3.7 наведено результати. Візьмемо для прикладу Таблицю 3.6, де повідомлення у відповідь має такий вигляд: [’10’: 4, ’11’: 2, ’01’: 10, ’00’: 84] Ці бінарні рядки представляють позиції вузлів у квантових блуканнях, тоді як пов’язані з ними значення вказують на кількість їх появ. Стани 10, 11, 01 та 00 використовуються для позначення кроків в алгоритмах квантових блукань.

Таблиця 3.5. Повідомлення у відповідь на симуляторі IBM для алгоритму квантових блукань.

Рівні оптимізації / Кроки	1	2	3	4
1	['00': 100]	['00': 100]	['00': 100]	['00': 100]
...	...	...	...	...
Загальна довжина повідомлення (у байтах)	113	122	124	124

Таблиця 3.6. Повідомлення на процесорі IBM CPU для алгоритму квантових блукань.

Рівні оптимізації / Кроки	1	2	3	4
1	['10': 4, '11': 2, '01': 10, '00': 84]	['11': 1, '01': 6, '10': 6, '00': 87]	['00': 100]	['00': 100]
...	...	...	...	...
Загальна довжина повідомлення (у байтах)	52	102	102	162

Таблиця 3.7. Повідомлення у відповідь для алгоритму квантових блукань на платформі Amazon Braket.

Алгоритми	На симуляторі	На QPU
Квантові блукання	[1: 0.512, 3: 0.488], [0: 0.54, 2: 0.463], [3: 1.0], [2: 1.0]	[1: 0.512, 3: 0.488], [0: 0.537, 2: 0.463], [3: 1.0], [2: 1.0]
Загальна довжина повідомлення (у байтах)	90	90

У Таблиці 3.7 результати подано у вигляді розподілу ймовірностей за позиціями. Наприклад: [1: 0.512, 3: 0.488] Це вказує на те, що після виконання певного кроку ймовірність перебування блукача (walker) у вузлі 1 становить 51,2.

Таблиця 3.8. Порівняльний аналіз тематичних досліджень затримки комунікації.

Характеристика	Алгоритм Гровера	Алгоритм Шора	Алгоритм квантових блукань
Тип алгоритму	Амплітудне підсилення	Пошук періоду / факторизація цілих чисел	Ймовірнісний обхід / пошук на графі
Застосування	Неструктурований пошук	Криптографічний аналіз	Моделювання на основі графів, оптимізація
Використані платформи	IBM Quantum, Amazon Braket	IBM Quantum, Amazon Braket	IBM Quantum, Amazon Braket
Формат повідомлення запиту	{name: 'grover', n: 110}	{name: 'shors', a: 7, N: 15}	{name: 'quantumwalk', n: 4, step: 4}
Розмір повідомлення запиту	27 байт	39 байт	42 байти
Кількість використаних кубітів	3 кубіти	8 кубітів	2–3 кубіти (для графа з 4 вузлами)
Формат відповіді (IBM)	Частота появи квантових станів (наприклад, {'110': 95, '000': 3})	Частота появи 8-бітних станів (наприклад, {'10000000': 27})	Частота появи 2-бітних станів (наприклад, {'00': 84, '01': 10})
Формат відповіді (Amazon)	Вектор ймовірностей для 8 станів (наприклад, [0.76 для '110'])	Подібні результати на основі частоти	Карта ймовірностей з індексацією за вузлами (наприклад, {1: 0.512, 3: 0.488})
Обсяг вихідних даних	Від малого до середнього	Від середнього до великого	Від малого до середнього

Поєднання цих форматів відповідей на різних платформах дозволяє детально оцінити комунікаційну затримку, особливо у зв’язку зі складністю квантового стану та розміром вихідних даних. Це тематичне дослідження наголошує на тому, як алгоритм квантових блукань — особливо у багатокрокових виконаннях — може генерувати різні обсяги вихідних даних, що впливає на профіль затримки.

Експериментальна установка демонструє важливість врахування специфічних для кожного алгоритму характеристик виводу під час бенчмаркінгу гібридних квантово-хмарних систем. Порівняльне резюме тематичних досліджень комунікаційної затримки наведено в Таблиці 3.8.

3.4. Вивчення затримки транспіляції

Щоб зробити результати порівнянними, час трансляції (transpilation) фіксується за допомогою Python, а не на основі часових міток, наданих серверами платформи. Такий підхід забезпечує точний контроль над збором даних під час усіх експериментів. Важливо зазначити, що Amazon Braket виконує трансляцію як процес «чорного ящика», не надаючи доступу до параметрів трансляції та можливості їх налаштування.

Відповідно, для аналізу на платформі Braket затримка трансляції фіксується як нульова, що відображає відсутність параметрів оптимізації, які можна було б налаштувати. Через поточну недоступність квантових процесорів у кастомізованій периферійній хмарі (edge cloud), усі експерименти проводилися на основній хмарній інфраструктурі, наданій IBM та Amazon.

Таблиця 3.9. Час транспіляції (в секундах) на платформі IBM з різними рівнями оптимізації алгоритму Гровера.

Платформа / Рівні оптимізації	Рівень 1	Рівень 2	Рівень 3	Рівень 4
На симуляторі	2.3×10^{-2}	1.5×10^{-2}	4.7×10^{-2}	2.0×10^{-2}
На QPU	1.29×10^{-1}	2.1×10^{-2}	2.2×10^{-2}	2.1×10^{-2}

У Таблиці 3.10 зафіксовано час трансляції (у секундах) для алгоритму Шора на симуляторі IBM та QPU за різних рівнів оптимізації. Таблиці 3.11 і 3.12 показують час трансляції (у секундах) для різної кількості кроків на платформі IBM.

Таблиця 3.10. Час транспіляції (в секундах) на IBM з різними рівнями оптимізації алгоритму Шора.

Платформа / Рівні оптимізації	Рівень 1	Рівень 2	Рівень 3	Рівень 4
На симуляторі	1.25	2.45×10^{-1}	01.06	1.52
На QPU	2.49×10	2.47×10	8.28×10	1.24×10^{2}

Таблиця 3.11. Час транспіляції (за секунди) на симуляторі алгоритму квантових блукань від IBM.

Кроки / Рівні оптимізації	Рівень 1	Рівень 2	Рівень 3	Рівень 4
1	01.02	6.0×10^{-2}	6.3×10^{-2}	6.6×10^{-2}
2	8.9×10^{-2}	9.8×10^{-2}	1.09×10^{-1}	1.22×10^{-1}
3	9.6×10^{-2}	9.5×10^{-2}	1.12×10^{-1}	1.08×10^{-1}
4	1.18×10^{-1}	9.9×10^{-2}	1.11×10^{-1}	1.1×10^{-1}

Таблиця 3.12. Час транспіляції (в секундах) на квантовому процесорі IBM для алгоритму квантових прогулянок.

Кроки / Рівні оптимізації	Рівень 1	Рівень 2	Рівень 3	Рівень 4
1	5.0×10^{-2}	2.7×10^{-2}	3.4×10^{-2}	4.3×10^{-2}
2	6.3×10^{-2}	6.3×10^{-2}	1.3×10^{-1}	1.17×10^{-1}
3	1.97×10^{-1}	6.3×10^{-2}	8.5×10^{-2}	1.1×10^{-1}
4	8.2×10^{-2}	1.19×10^{-1}	2.87×10^{-1}	1.22×10^{-1}

Тим не менш, оскільки тести виконуються з використанням як симуляторів, так і реальних QPU на їхніх відповідних нативних платформах, отримані результати створюють обґрунтовану та інформативну базу для порівняльної оцінки продуктивності. У Таблиці 3.9 зафіксовано затримку трансляції, виміряну в секундах, для алгоритму Гровера на симуляторі IBM та QPU за різних рівнів оптимізації.

3.5. Робота із затримкою виконання обудованого фрейморку

Етап виконання оцінюється з використанням двох середовищ симуляції: qasm_simulator на IBM Quantum та локального симулятора на Amazon Braket. Кожен експеримент проводиться зі 100 запусками (shots) для забезпечення статистичної значущості, що пояснює, чому загальна кількість появ у таблицях результатів завжди дорівнює 100. Через обмеження поточних API, точність (fidelity) у реальному часі неможливо було зафіксувати безпосередньо під час виконання. Натомість значення точності базуються на опублікованих бенчмарках із попередніх досліджень. Зокрема, припускається, що QPU IBM працюють із середньою точністю 94.98% [65], тоді як Amazon Braket повідомляє про точність 99.92% [68].

У симульованому середовищі точність приймається за 100% згідно з припущенням про систему без шумів [1], що дозволяє чіткіше оцінити обчислювальну продуктивність без помилок, спричинених шумом. Згідно з цим експериментальним налаштуванням, вимірюється затримка виконання, а три обрані квантові алгоритми обчислюються та порівнюються на обох платформах. Детальний аналіз результатів для цих алгоритмів наведено в наступних підрозділах, що дає уявлення про специфічну для кожної платформи продуктивність та обчислювальні вимоги кожного алгоритму. У Таблиці 3.13 зафіксовано час виконання на симуляторі та QPU IBM. У Таблиці 3.14 зафіксовано час виконання на симуляторі та QPU Amazon..

Таблиця 3.13. Час виконання (в секундах) алгоритму Гровера на платформі IBM із різними рівнями оптимізації.

Платформа / Рівні оптимізації	Рівень 1	Рівень 2	Рівень 3	Рівень 4
На симуляторі	8.7×10^{-2}	6.4×10^{-2}	1.8×10^{-2}	3.2×10^{-2}
На QPU	2.37×10^{-2}	3.66×10^{-2}	2.98×10^{-2}	2.52×10^{-2}

Таблиця 3.14. Час виконання (у секундах) алгоритму Гровера на Amazon.

Середовище виконання	Час виконання (у секундах)
На симуляторі	4.86×10^{-5}
На QPU	1.23×10^{2}

У таблицях 3.15 та 3.16 додатково показано час виконання алгоритму Шора в секундах на платформах IBM та Amazon відповідно.

Таблиця 3.15. Час виконання (у секундах) на платформі IBM з різними рівнями оптимізації для алгоритму Шора.

Платформа / Рівні оптимізації	Рівень 1	Рівень 2	Рівень 3	Рівень 4
На симуляторі	1.48×10^{-1}	1.89×10^{-1}	2.37×10^{-1}	1.77×10^{-2}
На QPU	9.45×10	6.17×10	7.41×10	8.09×10

Таблиця 3.16. Час виконання (у секундах) на платформі Amazon для алгоритму Шора.

Середовище виконання	Час виконання (у секундах)
На симуляторі	3.30×10^{-1}
На QPU	1.32×10

У таблицях 3.17–3.19 показано час виконання різних кроків на секундах на платформах IBM і Amazon відповідно.

Таблиця 3.17. Час виконання (у секундах) на симуляторі алгоритму квантових блукань від IBM.

Кроки / Рівні оптимізації	Рівень 1	Рівень 2	Рівень 3	Рівень 4
1	3.1×10^{-3}	4.0×10^{-3}	1.7×10^{-3}	1.8×10^{-3}
2	2.87×10^{-2}	2.1×10^{-3}	4.5×10^{-3}	2.7×10^{-3}
3	2.5×10^{-3}	2.1×10^{-3}	2.5×10^{-3}	1.29×10^{-2}
4	2.1×10^{-3}	2.29×10^{-2}	7.8×10^{-3}	3.0×10^{-3}

Таблиця 3.18. Час виконання (в секундах) алгоритму квантових прогулянок на процесорі QPU IBM.

Кроки / Рівні оптимізації	Рівень 1	Рівень 2	Рівень 3	Рівень 4
1	3.07×10	3.05×10	2.59×10	3.05×10
2	3.28×10	3.65×10	3.08×10	3.09×10
3	3.00×10	3.69×10	3.00×10	2.46×10
4	2.56×10	3.83×10	3.07×10	3.67×10

Таблиця 3.19. Час виконання (у секундах) алгоритму квантових прогулянок на Amazon.

Середовище виконання	Час виконання (у секундах)
На симуляторі	3.18×10^{-1}
На QPU	1.56×10

Показник затримки розраховується за такою формулою (3.3). У таблиці 3.20 представлені показники затримки, розраховані на платформах IBM Quantum та Amazon Quantum, що охоплюють як симулятори, і QPU.

Таблиця 3.20. Метод А — Показник затримки платформи IBM та Amazon (в секундах).

Алгоритми	IBM Simulator	IBM QPU	Amazon Simulator	Amazon QPU
Grover's	1.37×10^{-1}	1.47×10	1.08×10^{-2}	1.23×10^{2}
Shor's	2.89	2.47×10^{2}	3.71×10^{-1}	1.34×10
Quantum Walks	3.81×10^{-1}	1.66×10	3.37×10^{-1}	1.56×10

До цього часу дані щодо часу трансляції та виконання були зафіксовані та побудовані на рисунках 3.4–3.7, що подані нижче. Очевидно, що їхні тенденції загалом не є послідовними. Це означає, що порядкові значення (на рисунках вони позначені як i = {1, 2, 3, 4), використані у формулах (3.9) та (3.12), можуть призвести до систематичної похибки при поясненні трендів даних. Таким чином, для вирішення цієї проблеми пропонується модель Методу В, яка передбачає спочатку коригування значень j з подальшим оновленням значень затримки у формулах (3.10) та (3.13).

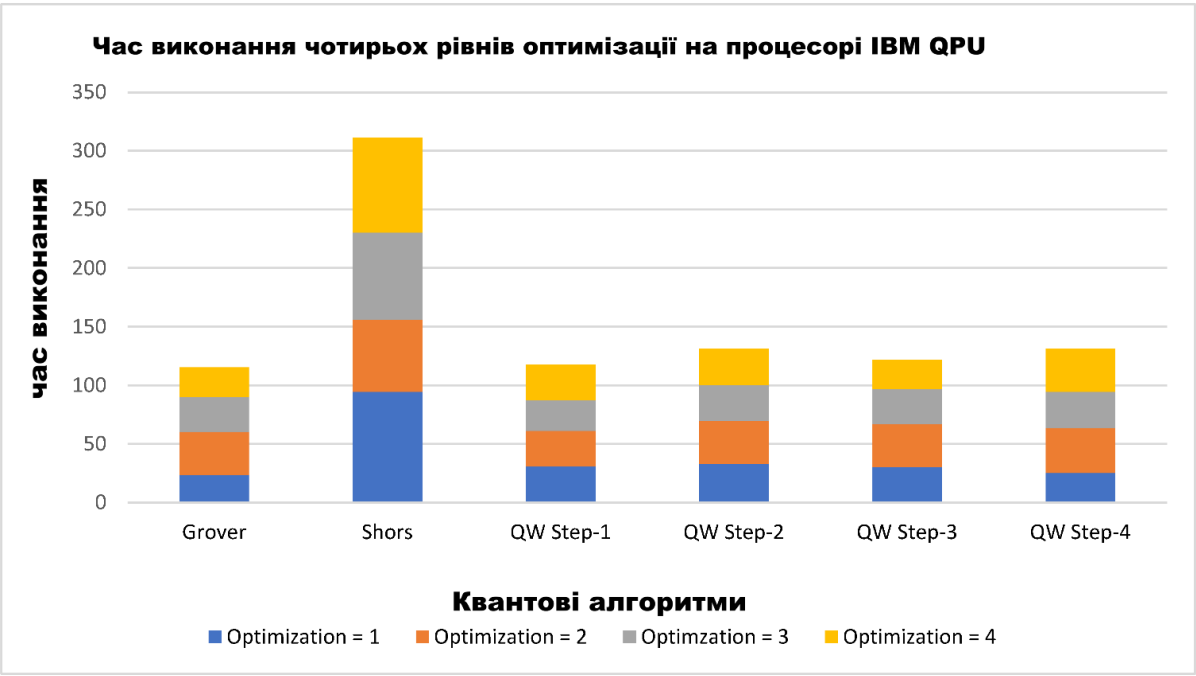

Рисунок 3.4 – Час виконання різних рівнів оптимізації на процесорі QPU IBM. Алгоритм Шора переважає інші алгоритми.

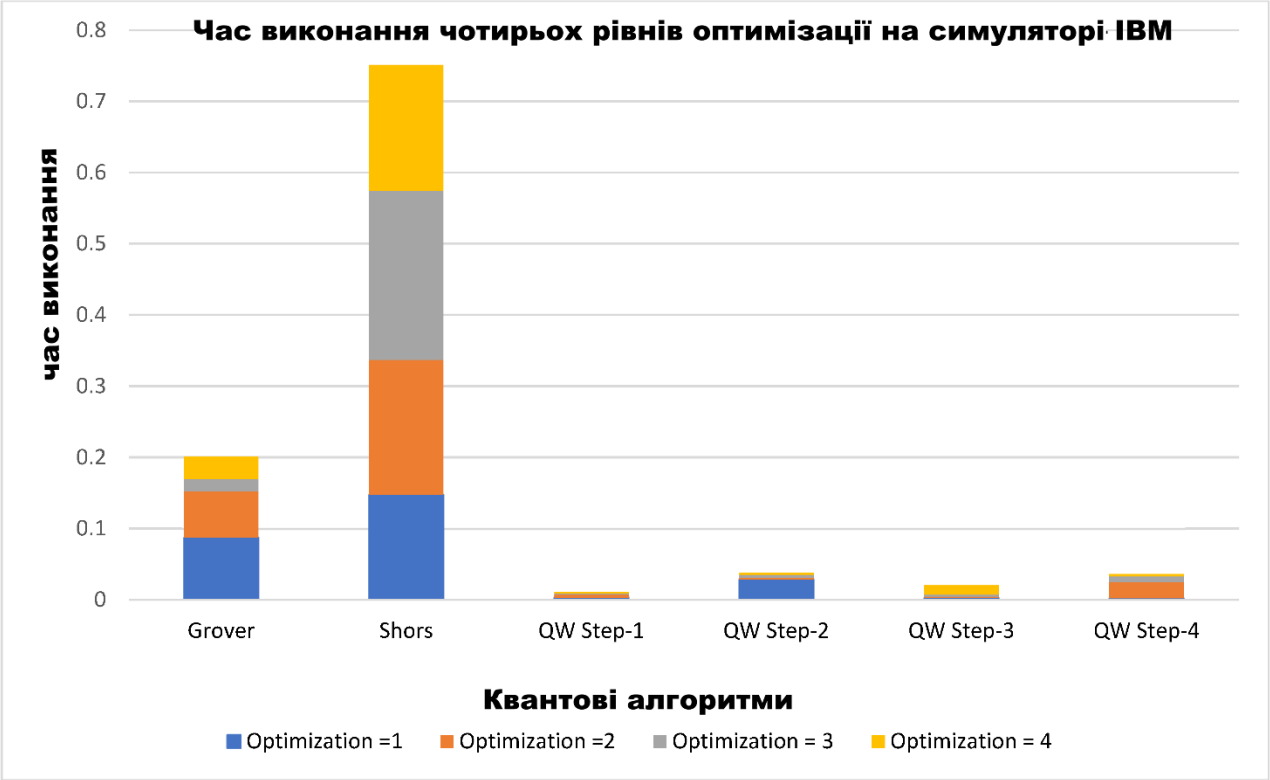

Рисунок 3.5 – Час виконання різних рівнів оптимізації на симуляторі IBM. Алгоритм Шора перевершує інші алгоритми.

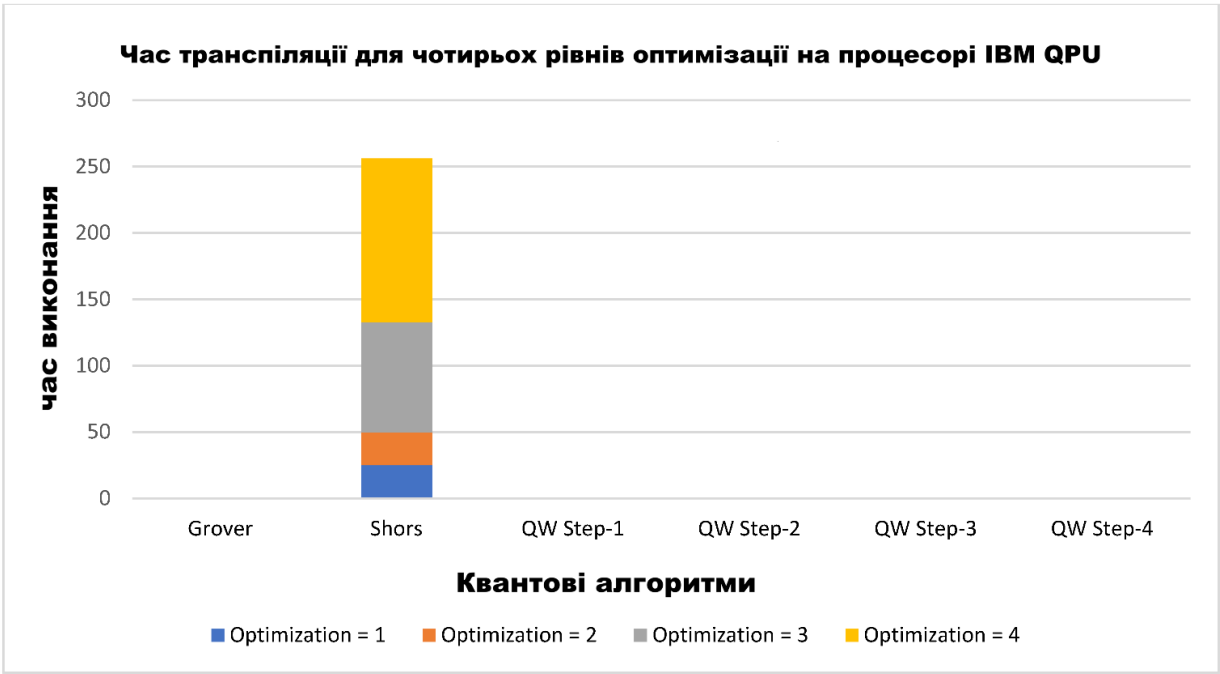

Рисунок 3.6 – Час транспіляції за різних рівнів оптимізації на процесорі IBM QPU. Алгоритм Шора перевершує інші алгоритми. В інших значення дуже низькі.

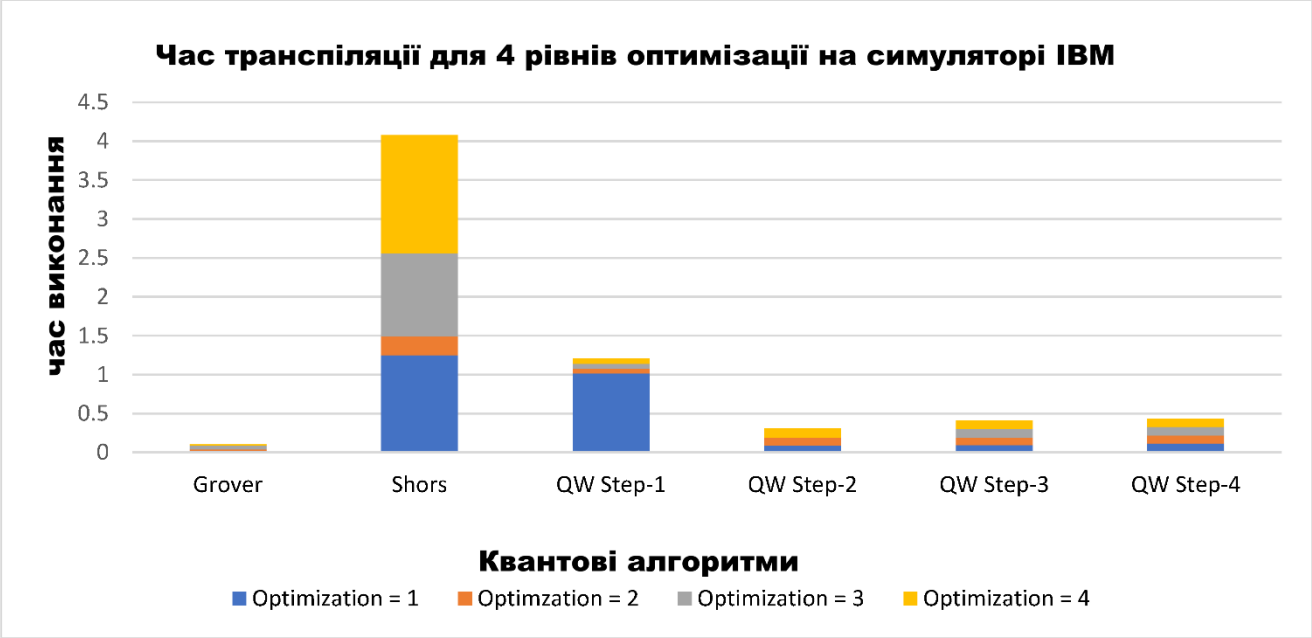

Рисунок 3.7 – Час транспіляції за різних рівнів оптимізації на симуляторі IBM. Алгоритм Шора перевершує інші алгоритми.

Оцінка затримки розраховується на основі формули (3.3). У Таблиці 3.21 представлені оцінки затримки, розраховані для платформ IBM Quantum та Amazon Quantum, що охоплюють як симулятори, так і QPU.

Таблиця 3.21. Метод B — Показник затримки платформи IBM та Amazon (в секундах).

Алгоритми	IBM Simulator	IBM QPU	Amazon Simulator	Amazon QPU
Grover's	1.59×10^{-1}	2.38×10	1.08×10^{-2}	1.23×10^{2}
Shor's	1.20	3.29×10^{2}	3.71×10^{-1}	1.34×10
Quantum Walks	1.32×10^{-1}	2.71×10	3.37×10^{-1}	1.56×10

3.6. Аналіз та обговорення результатів тестування

1. Аналіз впливу рівнів оптимізації компілятора на часову складність виконання В інструментарії хмарного середовища IBM Quantum Lab реалізовано чотири рівні оптимізації компілятора (у межах даного дослідження позначені як параметр $optimization = \{1,2,3,4\}$, що відповідає системним рівням 0–3). Оцінка їхнього впливу на продуктивність обчислень як у середовищі емуляції (програмних симуляторах), так і на фізичних квантових процесорах (QPU) виявила нелінійну залежність. Згідно з експериментальними даними, наведеними на Рисунках 3.8 та 3.9, ефект від застосування вищих рівнів оптимізації суттєво варіюється залежно від математичної структури алгоритму. Це підтверджує гіпотезу про те, що ефективність низькорівневих стратегій компіляції квантового ПЗ є алгоритмозалежною детермінантою і не може бути уніфікована для всіх класів обчислювальних завдань.

2. Порівняльний аналіз продуктивності програмних симуляторів та фізичних QPU Емпіричні результати свідчать, що програмні емулятори на обох досліджуваних платформах (IBM та Amazon Braket) стабільно перевершують за продуктивністю відповідні фізичні QPU. Дана тенденція чітко простежується через мінімізацію інтегральної метрики затримки (Latency Score) в межах обох розроблених математичних моделей оцінювання — Методу А та Методу В (див. Таблиці 3.20 і 3.21). Головним деструктивним фактором сучасних фізичних QPU залишається обмежена кубітна потужність та високий рівень квантового шуму, що на поточному етапі розвитку технологій нівелює можливість досягнення практичної квантової переваги (Quantum Advantage) для складних систем. Отримані висновки корелюють із сучасними літературними джерелами: архітектури QPU, здатні забезпечити стійке функціонування складних промислових рішень (зокрема, безпечних гетерогенних периферійно-хмарних інфраструктур — Hybrid Edge-Cloud), з’являться на ринку не раніше ніж через 5 років.

3. Оцінка накладних обчислювальних витрат на етапі трансляції квантових схем Процес трансляції (Transpilation), що полягає у топологічному відображенні абстрактної логічної квантової схеми на специфічний архітектурний граф зв'язності фізичного процесора, є критичною фазою життєвого циклу виконання квантових програм. Дослідження часових характеристик цього етапу (Рисунки 3.10 та 3.11) виявило його експоненційну залежність від розмірності простору станів (кількості використовуваних кубітів). Серед протестованого пулу алгоритмів найбільші часові витрати на трансляцію та безпосереднє виконання на обох платформах продемонстрував алгоритм Шора (Shor's algorithm). Це зумовлено тим, що він оперує 8 кубітами — максимальною кількістю в межах проведеного експерименту. Отриманий результат підтверджує стрімке зростання обчислювальних накладних витрат (Computational Overhead) при масштабуванні квантових схем та актуалізує задачу розробки оптимізованих алгоритмів попереднього топологічного аналізу.

4. Вплив рівнів компіляції на вірність відтворення станів (Fidelity) Експериментально доведено, що зміна рівнів оптимізації компілятора IBM (від 0 до 3) забезпечує лише незначне (маргінальне) покращення показника вірності результатів (Fidelity), проте спричиняє радикальні девіації часової складності обчислень. Дане явище підкреслює домінантну роль механізмів компіляції у формуванні інтегральної ефективності квантового програмного забезпечення. Для підвищення точності метричного аналізу в роботі запропоновано модифікований метод бенчмаркінгу, який інтегрує обраний рівень оптимізації як зважений коефіцієнт (Weighted Factor) у загальну модель оцінки системної продуктивності та алгоритмічної ефективності.

5. Системна оцінка кросплатформеної обчислювальної ефективності Незважаючи на те, що фізичні процесори IBM Quantum володіють більшою номінальною кількістю кубітів, хмарна платформа Amazon Braket продемонструвала вищу продуктивність під час виконання двох із трьох тестових алгоритмів (див. підсумкові Таблиці 3.20 і 3.21). Детермінантами такої розбіжності є архітектурні особливості топологій процесорів, внутрішні низькорівневі стратегії

оптимізації вендорних компіляторів та ступінь їхньої сумісності з логічною структурою конкретних алгоритмів. Отримані висновки спростовують поширений екстенсивний підхід в інженерії квантового ПЗ, згідно з яким суха кількість кубітів є єдиним критерієм потужності платформи. Доведено необхідність переходу до комплексного (мультикритеріального) оцінювання, що обов’язково має включати аналіз методів придушення помилок (Error Mitigation), алгоритмів оптимізації схем та загальний рівень інтеграції програмно-апаратних засобів.

4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Ергономічні проблеми безпеки життєдіяльності.

Ергономіка є міждисциплінарною наукою, яка комплексно вивчає особливості взаємодії людини з технічними системами, робочим середовищем та інформаційними потоками. Головна мета ергономічного підходу полягає у створенні таких умов праці, які максимально враховують фізіологічні, психологічні, анатомічні та когнітивні особливості людини, забезпечуючи високий рівень безпеки, ефективності та комфорту у трудовій діяльності.

Сам термін походить від грецьких слів ergon (праця) та nomos (закон) і був уперше запропонований польським ученим Войцехом Ястшембовським у 1857 році. У сучасному контексті інженерії та проектування ергономіка виконує фундаментальну роль у системі безпеки життєдіяльності, оскільки правильна організація робочих систем безпосередньо впливає на зниження виробничих ризиків, профілактику професійних захворювань і загальний добробут працівників [42]. До основних завдань ергономіки належить проєктування людино-машинних систем із чітким урахуванням людських можливостей та природних обмежень, а також оптимізація інтерфейсів і органів керування задля забезпечення їх інтуїтивного й безпечного використання.

Важливе місце посідає мінімізація фізичного та психоемоційного навантаження на працівника шляхом адаптації робочого простору до індивідуальних антропометричних, фізіологічних і когнітивних характеристик людини. Усе це в сукупності дозволяє суттєво підвищити загальну продуктивність праці завдяки зниженню стомлюваності та створенню оптимального мікроклімату й комфортних умов. Нормативно-правове регулювання та стандартизація в цій галузі спираються на комплекс національних та міжнародних документів. Зокрема, загальні ергономічні принципи, які необхідно враховувати під час проєктування безпечних машин і робочих місць, регламентуються стандартом ДСТУ EN 614-1:2018 [43]. Загальні вимоги щодо інтеграції ергономічних факторів у структуру робочого середовища встановлює ДСТУ ISO 6385:2018 [44].

Для технічного проєктування, що враховує фізичні параметри населення, базові антропометричні вимірювання людини стандартизовано в ДСТУ ISO 7250-1:2018. Окремі аспекти виробничого середовища контролюються будівельними та санітарними нормами, серед яких ДБН В.2.5-28:2018 визначає параметри природного і штучного освітлення робочих зон [45], СанПіН 3.3.2.007-98 встановлює правила безпечної роботи з відеотерміналами [46], а СанР 173–96 обмежує гранично допустимі рівні шуму на робочих місцях [47].

Державні гарантії та загальні правові засади організації безпечних умов праці додатково закріплені в Законі України «Про охорону праці» та Кодексі законів про працю України. Ефективне проєктування робочого простору базується на низці ключових ергономічних принципів. Першочерговим є орієнтація на людину, що передбачає пристосування умов праці до працівника, а не навпаки. Це вимагає строгої антропометричної відповідності, коли розміри меблів та обладнання чітко корелюють із фізичними параметрами користувача. Усі елементи керування та інструменти мають розміщуватися в межах зон зручної досяжності для мінімізації зайвих рухів. Важливу роль відіграє інформаційна ергономіка, згідно з якою будь-яка інформація на екранах або у вигляді сигналів має бути легкою для сприйняття, логічною та однозначною.

Для зниження зорового навантаження екран повинен мати оптимальний кут нахилу та яскравість, а рівень освітленості має становити не менше 300 лк. Психологічний комфорт забезпечується створенням спокійного, передбачуваного середовища без дратівливих факторів. Зниження загального навантаження досягається через чергування видів діяльності, обов’язкові регламентовані перерви та зміну положення тіла. Також система повинна мати можливості індивідуального налаштування меблів і параметрів приміщення.

Безпечна взаємодія з технікою гарантується відсутністю ризику травм, наявністю систем аварійного вимикання та чітких інструкцій, а сприяння відновленню сил забезпечується облаштуванням зон відпочинку, належною вентиляцією та озелененням. Нехтування зазначеними ергономічними принципами призводить до серйозних негативних наслідків як для працівника, так і для

виробничого процесу загалом. До них належать професійні захворювання, такі як остеохондроз, варикозне розширення вен, тунельний синдром та короткозорість. Окрім цього, суттєво підвищується рівень помилок і виникають передумови для аварійних ситуацій через психоемоційне виснаження, стрес та депресивні стани працівників.

Як наслідок, різко знижуються продуктивність праці та концентрація уваги, що створює високі ризики травматизму, особливо в умовах інтенсивної або монотонної діяльності. Таким чином, ергономіка є невід'ємним елементом загальної системи безпеки, що охоплює організацію праці, проєктування робочого місця та взаємодію людини з технічними засобами та інформаційним середовищем. Її головною метою залишається зниження професійних ризиків, підвищення ефективності діяльності та формування здорового, безпечного робочого середовища. Суворе дотримання принципів ергономіки є обов’язковим як на етапі інженерного проєктування систем, так і в процесі безпосередньої експлуатації робочого простору.

4.2 Значення автоматизації виробничих процесів в питаннях охорони праці

Автоматизація виробничих процесів у сфері охорони праці передбачає активне впровадження сучасних технічних засобів, які забезпечують безперервний контроль, глибокий аналіз та оперативне реагування на будь-які відхилення в режимах роботи виробничого обладнання, умовах праці та поведінці персоналу. Головною метою таких заходів є ефективне запобігання аварійним ситуаціям, нещасним випадкам і мінімізація ризиків виникнення професійних захворювань. Відповідно до статті 13 Закону України «Про охорону праці», на роботодавця покладається обов’язок впроваджувати прогресивні засоби виробництва та передові технології з обов'язковим урахуванням вимог безпеки, гігієни праці та охорони здоров’я.

Водночас системна автоматизація повинна інтегруватися не лише безпосередньо у виробничі процеси, а й у загальну структуру управління охороною праці, формуючи єдиний цілісний комплекс технічних та організаційних заходів [48]. Проведені наукові дослідження, зокрема праці А. П. Бочковського та Н. Ю. Сапожнікової (2017), доводять, що переважна більшість виробничих травм в Україні зумовлена причинами організаційного (67%) та психофізіологічного (21%) характеру. Серед ключових деструктивних факторів науковці виділяють систематичне невиконання інструкцій з охорони праці, перебування працівників у потенційно небезпечних зонах, а також незадовільний технічний стан експлуатованого обладнання.

Задля мінімізації негативного впливу так званого людського фактора пропонується масштабне впровадження систем автоматизованого контролю та підвищення безпеки промислових виробництв. До базового складу таких комплексів зазвичай входять високоточні датчики руху, деформації, температури й вологості, системи інтелектуального відеоспостереження з фіксацією дій персоналу, а також електронні термінали для верифікації на вході до системи й жорсткого контролю доступу до складного обладнання. Крім того, невід'ємними компонентами є механізми автоматичного блокування небезпечних ділянок чи устаткування, засоби світло-звукової сигналізації тривоги та централізовані електронні бази даних, у яких фіксуються виявлені порушення, проходження інструктажів і результати обов'язкових медичних оглядів [49].

Практичне впровадження подібних автоматизованих систем дозволяє суттєво зменшити кількість нещасних випадків завдяки попередженню порушень у режимі реального часу, а також кардинально підвищити рівень контролю за поточним технічним станом будівель, споруд та інженерного устаткування. Разом із цим забезпечується можливість ведення чіткого персоніфікованого обліку знань, пройдених інструктажів і реального дотримання правил охорони праці, що дозволяє формувати індивідуальні навчальні модулі для цілеспрямованого підвищення професійної компетентності персоналу та своєчасно сигналізувати про виникнення будь-яких небезпечних ситуацій чи відхилень від встановлених

нормативів. Окрім цього, автоматизація дає змогу повноцінно реалізувати системний підхід до безпеки, регламентований стандартом ДСТУ EN ISO 12100:2016 щодо загальних принципів проєктування, оцінки та зменшення ризиків [50], а також повністю відповідає сучасним вимогам ДСТУ EN ISO 45001:2019, який є міжнародним еталоном для систем управління охороною праці та безпекою на робочих місцях [51-53].

У сучасне виробниче середовище такі інтелектуальні системи інтегруються за допомогою програмованих логічних контролерів (ПЛК), розгалужених SCADA-систем, сенсорних мереж та передових методів аналітики великих даних, що дозволяє виявляти негативні тенденції у поведінці техніки чи персоналу задовго до того, як вони призведуть до реальної аварії.

Таким чином, автоматизація охоплює створення комплексної адаптивної системи забезпечення безпечних умов праці, яка гармонійно поєднує в собі процеси безперервного моніторингу, глибокого аналізу, оперативного управління та превентивного навчання. Це повністю відповідає визнаному міжнародному підходу до управління ризиками в промисловій безпеці, орієнтованому не на ліквідацію наслідків, а на випередження та усунення першопричин виникнення небезпек. У контексті сучасного високотехнологічного виробництва автоматизація є критичною необхідністю для надійного збереження життя, здоров’я та працездатності людського капіталу.

ВИСНОВКИ

У результаті виконання дипломної роботи було проведено комплексне дослідження та отримано вагомі науково-практичні результати в галузі інженерії програмного забезпечення. У межах практичної реалізації було розроблено та представлено інноваційний фреймворк із відкритим вихідним кодом у вигляді бібліотеки Python для бенчмаркінгу квантових комп'ютерів на базі вентилів. Ця бібліотека здатна ефективно тестувати пристрої NISQ шляхом перевірки їхньої здатності розрізняти два вимірювання фон Неймана.

Для зручності взаємодії створено спрощений, готовий до використання інтерфейс командного рядка (CLI), який дозволяє запускати тести із застосуванням заздалегідь визначеного параметризованого Фур'є-сімейства вимірювань. Для більш складних сценаріїв у фреймворку передбачено можливість гнучкого застосування вимірювань, визначених безпосередньо користувачем, замість стандартних конфігурацій. Крім того, у роботі детально досліджено інтеграцію технологій квантових обчислень у класичні периферійно-хмарні мережі з використанням загальнодоступних хмарних платформ, таких як IBM Quantum Lab та Amazon Braket.

Шляхом впровадження відомих квантових алгоритмів та використання показників затримки як основної метрики ефективності було проведено повну порівняльну оцінку симуляторів та квантових процесорів (QPU) за різних рівнів оптимізації та конфігурацій систем. На основі проведених експериментів було сформовано кілька ключових висновків. Встановлено, що в усіх досліджених сценаріях квантові симулятори демонструють кращі показники затримки порівняно з фізичними квантовими процесорами. Цей результат чітко підкреслює поточні обмеження технологічної зрілості сучасних QPU, зокрема щодо їхньої масштабованості, надійності та стійкості до зовнішніх шумів. Також виявлено, що хоча компанія IBM пропонує більшу кількість кубітів, платформа Amazon досягла вищої продуктивності для всіх протестованих алгоритмів. Цей висновок доводить, що архітектура системи, ефективність компілятора та методологии оптимізації

мають значно більший вплив на кінцевий результат, ніж сухі апаратні характеристики, а сама по собі кількість кубітів є недостатньою мірою реальних можливостей платформи.

Додатково експерименти показали, що процес транспіляції створює значні накладні витрати, які можуть негативно вплинути на загальний час виконання завдань. Це підкреслює гостру необхідність досягнення компромісного балансу між стратегіями оптимізації під час транспіляції та практичними обмеженнями реального виконання обчислень. Результати демонструють високий трансформаційний потенціал квантових обчислень у периферійно-хмарних середовищах і закладає фундаментальну основу для майбутніх наукових розробок. Оскільки більшість поточних експериментів базуються на припущенні про доступність квантово-класичних ресурсів на хмарних платформах, що повністю узгоджується з політикою Amazon та IBM, для емуляції периферійної обробки обмежена попередня обробка виконувалася локально шляхом симуляції квантових процесорів на класичних CPU. Це дозволило успішно виконати обрані алгоритми та виміряти наскрізні затримки.

У майбутньому дослідження буде розширено у напрямку складніших розподілених середовищ із впровадженням гібридної обробки на всіх існуючих рівнях архітектури. У межах подальшого розвитку визначено кілька перспективних напрямків дослідження. Першим кроком планується проведення глибшого вивчення взаємодії між рівнями оптимізації транспіляції та топологією QPU для чіткого прояснення компромісів між агресивною оптимізацією та накладними витратами на виконання. Також включення ширшого спектра квантових алгоритмів дозволить суттєво уточнити формули тестування, оновити ключі параметри та отримати розширене розуміння повної поведінки платформи під час виконання різноманітних обчислювальних завдань. Разом з тим, очікуваний прогрес у розробці практичних QPU дозволить реалізувати гнучкий розподіл кубітів, що дасть змогу завданням динамічно використовувати обчислювальну потужність відповідно до рівня складності конкретної задачі. Наступним етапом стане розширення розробленого фреймворку з класичних на квантові

комунікаційні мережі, що забезпечить отримання більш точного розуміння справжньої наскрізної продуктивності в гібридних квантово-периферійно-хмарних системах. Наостанок, впровадження моніторингу точності в реальному часі замість статичних значень точності покращить достовірність оцінки продуктивності та посилить існуючі методології бенчмаркінгу. Усі ці напрямки сукупно спрямовані на просування практичної інтеграції квантових обчислень в архітектури периферійно-хмарних систем наступного покоління, що дозволить ефективно подолати розрив між експериментальними прототипами та масштабованими розгортаннями в реальному світі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Rigetti computing [Електронний ресурс]. – Режим доступу: https://www.rigetti.com/ (дата звернення: 18.05.2026).

2. IBM quantum [Електронний ресурс]. – Режим доступу: https://www.ibm.com/quantum (дата звернення: 18.05.2026).

3. Oxford quantum [Електронний ресурс]. – Режим доступу: http://oxfordquantum.org/ (дата звернення: 18.05.2026).

4. IonQ [Електронний ресурс]. – Режим доступу: https://ionq.com/ (дата звернення: 18.05.2026).

5. Xanadu [Електронний ресурс]. – Режим доступу: https://www.xanadu.ai/ (дата звернення: 18.05.2026).

6. D-wave systems [Електронний ресурс]. – Режим доступу: https://www.dwavesys.com/ (дата звернення: 18.05.2026).

7. QuEra [Електронний ресурс]. – Режим доступу: https://www.quera.com/ (дата звернення: 18.05.2026).

8. QCS documentation [Електронний ресурс]. – Режим доступу: https://docs.rigetti.com/qcs/ (дата звернення: 18.05.2026).

9. PyQuil documentation [Електронний ресурс]. – Режим доступу: https://pyquil-docs.rigetti.com/en/stable/ (дата звернення: 18.05.2026).

10. Qiskit [Електронний ресурс]. – Режим доступу: https://qiskit.org/ (дата звернення: 18.05.2026).

11. IBM quantum experience [Електронний ресурс]. – Режим доступу: https://quantum-computing.ibm.com/ (дата звернення: 18.05.2026).

12. Amazon braket [Електронний ресурс]. – Режим доступу: https://aws.amazon.com/braket/ (дата звернення: 18.05.2026).

13. Zapata orquestra platform [Електронний ресурс]. – Режим доступу: https://www.zapatacomputing.com/orquestraplatform/ (дата звернення: 18.05.2026).

14. McCaskey A. J. XACC: a system-level software infrastructure for heterogeneous quantum–classical computing / A. J. McCaskey, D. I. Lyakh, E. F. Dumitrescu, S. S. Powers, T. S. Humble // Quant Sci Technol. – 2020. – Vol. 5(2). – P. 024002.

15. CUDA quantum [Електронний ресурс] / The CUDA Quantum development team. – Режим доступу: https://github.com/NVIDIA/cuda-quantum (дата звернення: 18.05.2026).

16. Preskill J. Quantum computing in the NISQ era and beyond / J. Preskill // Quantum. – 2018. – Vol. 2. – P. 79.

17. Preskill J. Quantum computing 40 years later / J. Preskill // arXiv preprint arXiv:2106.10522. – 2021.

18. Benchmarking near-term quantum computers via random circuit sampling / Y. Liu, M. Otten, R. Bassirianjahromi, L. Jiang, B. Fefferman // arXiv preprint arXiv:2105.05232. – 2021.

19. Randomized benchmarking of quantum gates / E. Knill, D. Leibfried, R. Reichle, J. Britton, R. B. Blakestad, J. D. Jost [et al.] // Phys Rev A. – 2008. – Vol. 77(1). – P. 012307.

20. Wallman J. J. Randomized benchmarking with confidence / J. J. Wallman, S. T. Flammia // New J Phys. – 2014. – Vol. 16(10). – P. 103032.

21. General framework for randomized benchmarking / J. Helsen, I. Roth, E. Onorati, A. H. Werner, J. Eisert // PRX Quantum. – 2022. – Vol. 3(2). – P. 020357.

22. Cornelissen A. Scalable benchmarks for gate-based quantum computers / A. Cornelissen, J. Bausch, A. Gilyén // arXiv preprint arXiv:2104.10698. – 2021.

23. Validating quantum computers using randomized model circuits / A. W. Cross, L. S. Bishop, S. Sheldon, P. D. Nation, J. M. Gambetta // Phys Rev A. – 2019. – Vol. 100(3). – P. 032328.

24. Quantum optimization using variational algorithms on near-term quantum devices / N. Moll, P. Barkoutsos, L. S. Bishop, J. M. Chow, A. Cross, D. J. Egger [et al.] // Quantum Sci Technol. – 2018. – Vol. 3(3). – P. 030503.

25. Pelofske E. Quantum volume in practice: What users can expect from NISQ devices / E. Pelofske, A. Bärtschi, S. Eidenbenz // IEEE Trans Quantum Eng. – 2022. – Vol. 3. – P. 1–19.

26. Quetschlich N. MQT Bench: Benchmarking software and design automation tools for quantum computing / N. Quetschlich, L. Burgholzer, R. Wille // arXiv preprint arXiv:2204.13719. – 2022.

27. MQTBench [Електронний ресурс]. – Режим доступу: https://github.com/cda-tum/MQTBench (дата звернення: 18.05.2026).

28. SupermarQ: A scalable quantum benchmark suite / T. Tomesh, P. Gokhale, V. Omole, G. S. Ravi, K. N. Smith, J. Viszlai [et al.] // 2022 IEEE international symposium on high-performance computer architecture. – IEEE, 2022. – P. 587–603.

29. SupermarQ [Електронний ресурс]. – Режим доступу: https://github.com/SupertechLabs/SupermarQ (дата звернення: 18.02.2023).

30. Qiskit benchmarks [Електронний ресурс]. – Режим доступу: https://github.com/qiskit-community/qiskit-benchmarks (дата звернення: 18.05.2026).

31. Forest benchmarking: QCVV using PyQuil [Електронний ресурс]. – Режим доступу: https://github.com/rigetti/forest-benchmarking (дата звернення: 18.05.2026).

32. Chernyavskiy A. Y. Quantum tomography benchmarking / A. Y. Chernyavskiy, Y. I. Bogdanov // Quantum Inf Process. – 2021. – Vol. 20. – P. 1–20.

33. Quantum tomography benchmarking [Електронний ресурс]. – Режим доступу: https://github.com/PQCLab/pyQTB (дата звернення: 18.05.2026).

34. Experimental evaluation of nisq quantum computers: Error measurement, characterization, and implications / T. Patel, A. Potharaju, B. Li, R. B. Roy, D. Tiwari // SC20: International conference for high performance computing, networking, storage and analysis. – IEEE, 2020. – P. 1–15.

35. Qasmbench: A low-level quantum benchmark suite for nisq evaluation and simulation / A. Li, S. Stein, S. Krishnamoorthy, J. Ang // ACM Trans Quantum Comput. – 2023. – Vol. 4(2). – P. 1–26.

36. Quantum volume in practice [Електронний ресурс]. – Режим доступу: https://github.com/lanl/Quantum-Volume-in-Practice (дата звернення: 18.05.2026).

37. Characterizing quantum supremacy in near-term devices / S. Boixo, S. V. Isakov, V. N. Smelyanskiy, R. Babbush, N. Ding, Z. Jiang [et al.] // Nat Phys. – 2018. – Vol. 14(6). – P. 595–600.

38. Quantum supremacy using a programmable superconducting processor / F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends [et al.] // Nature. – 2019. – Vol. 574(7779). – P. 505–510.

39. Towards a quantum benchmark suite with standardized KPIs / C. Kai-Uwe Becker, N. Tcholtchev, I.-D. Gheorghe-Pop, S. Bock, R. Seidel, M. Hauswirth // 2022 IEEE 19th international conference on software architecture companion. – 2022. – P. 160–163.

40. Jałowiecki K. PyQBench supplemental materials [Електронний ресурс] / K. Jałowiecki, P. Lewandowska, Ł. Pawela. – Режим доступу: https://github.com/iitis/PyQBench/blob/master/supplemental.pdf (дата звернення: 18.05.2026).

41. Strategies for optimal single-shot discrimination of quantum measurements / Z. Puchała, Ł. Pawela, A. Krawiec, R. Kukulski // Phys Rev A. – 2018. – Vol. 98(4). – P. 042103.

42. Безпека життєдіяльності : навч. посібник / Т. Є. Стиценко, Г. В. Пронюк, Н. М. Сердюк, І. І. Хондак. – Харків : ХНУРЕ, 2018. – 336 с.

43. Електронне навчання ТНТУ. Курс “Безпеки життєдіяльності та охорони праці” [Електронний ресурс]. – 2025. – Режим доступу: https://dl.tntu.edu.ua/index.php

44. ДСТУ EN 614-1:2018. Безпека машин. Ергономічні принципи проєктування. Частина 1: Загальні принципи від 18.07.2018 № 226 [Електронний ресурс]. – 2018. – Режим доступу: https://online.budstandart.com/ua/catalog/doc-page.html?id_doc=77654

45. ДСТУ ISO 6385:2018. Основні принципи ергономіки при проєктуванні систем від 19.12.2018 № 513 [Електронний ресурс]. – 2018. – Режим доступу: https://online.budstandart.com/ua/catalog/doc-page.html?id_doc=80890

46. ДБН В.2.5-28:2018. Природне і штучне освітлення від 03.10.2018 № 264 [Електронний ресурс]. – 2018. – Режим доступу: https://online.budstandart.com/ua/catalog/doc-page.html?id_doc=79885

47. СанПіН 3.3.2.007-98. Гігієнічні вимоги до відеотерміналів персональних електронно-обчислювальних машин від 10.12.98 № 7 [Електронний ресурс]. – 2025. – Режим доступу: https://zakon.rada.gov.ua/rada/show/v0007282-98#Text

48. СанР 173–96. Гранично допустимі рівні шуму на робочих місцях від 22.02.2019 № 463 [Електронний ресурс]. – 2025. – Режим доступу: https://zakon.rada.gov.ua/laws/show/z0281-19#Text

49. Закон України «Про охорону праці» від 14.10.1992 № 2694-XII [Електронний ресурс] / Верховна Рада України. – 2025. – Режим доступу: https://zakon.rada.gov.ua/laws/show/2694-12#Text

50. Бочковський А. П. Формалізація системи автоматизованого контролю і підвищення безпеки виробництв / А. П. Бочковський, Н. Ю. Сапожнікова // Вісник Львівського державного університету безпеки життєдіяльності. – 2017. – № 15. – С. 114–123. – Режим доступу: http://nbuv.gov.ua/UJRN/Vldubzh_2017_15_17

51. Кодекс законів про працю України від 10.12.1971 № 322-VIII [Електронний ресурс] / Верховна Рада України. – 2025. – Режим доступу: https://zakon.rada.gov.ua/laws/show/322-08#Text

52. ДСТУ EN ISO 12100:2016. Безпечність машин. Загальні принципи проєктування. Оцінка ризику та зменшення ризику від 13.12.2016 № 426 [Електронний ресурс]. – 2016. – Режим доступу: https://online.budstandart.com/ua/catalog/doc-page?id_doc=54659

53. ДСТУ EN ISO 45001:2019. Системи управління охороною здоров’я та безпекою праці. Вимоги та настанови щодо застосування від 26 грудня 2019 року № 506 [Електронний ресурс]. – 2019. – Режим доступу: https://online.budstandart.com/ua/catalog/doc-page?id_doc=87509

ДОДАТКИ

Додаток А – Диск із кваліфікаційною роботою бакалавра