УДК 004.942+004.415:530.145 DOI: 10.31673/2412-4338.2025.038707

Йовчев Димитр Костадінович
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ
ORCID 0009-0005-0436-1414

СИМУЛЯЦІЇ КВАНТОВИХ АЛГОРИТМІВ ЗА ДОПОМОГОЮ БІБЛІОТЕЦІ QISKIT AER І МЕТОДУ МАТРИЧНОГО ДОБУТКУ СТАНІВ НА CPU ТА GPU

Анотація. *У цій статті представлено всебічне дослідження продуктивності симуляції квантових схем методом матричних добуткових станів (Matrix Product States, MPS) у середовищі Qiskit Aer. Основна увага зосереджена на порівнянні обчислювальної ефективності реалізацій на центральному процесорі (CPU) та графічному процесорі (GPU) для різних симуляційних платформ. Експериментально досліджено чотири ключові квантові алгоритми: Variational Quantum Eigensolver (VQE), Quantum Fourier Transform (QFT), Quantum Approximate Optimization Algorithm (QAOA) і Greenberger–Horne–Zeilinger (GHZ). Для кожного алгоритму побудовано графіки, що відображають залежність часу виконання від кількості кубітів і типу симулятора. Також наведено єдину діаграму для порівняння всіх чотирьох алгоритмів з використанням методу MPS у Qiskit Aer. Результати свідчать, що хоча симулятор MPS у Qiskit Aer наразі реалізовано лише для CPU, він перевищує традиційний симулятор станового вектора (statevector) для схем із низьким рівнем заплутаності. Водночас відсутність GPU-акселерації обмежує масштабованість MPS-симулятора. Аналіз показує, що впровадження апаратного прискорення SVD-операцій за допомогою бібліотек NVIDIA cuQuantum і cuTensorNet може забезпечити до семикратне прискорення. Також проведено порівняльний аналіз GPU-реалізацій у Qiskit Aer і CUDA-Q, включно з бібліотеками cuStateVec, LAPACK і cuSOLVER. Створено графіки, що демонструють сильні та слабкі сторони симуляторів і методів симуляції Qiskit Aer, зокрема, метод MPS залишається одним із найкращих за часом виконання CPU, але часом поступається іншим методам на GPU. Запропоновано напрями подальшої оптимізації MPS, зокрема розробку гібридної CPU/GPU-архітектури і використання сучасних GPU-бібліотек для прискорення важливих операцій.*

Ключові слова: *квантові обчислення, квантова симуляція, Qiskit Aer, MPS, вектор станів, cudaq.*

Yovchev Dymytr
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv
ORCID 0009-0005-0436-1414

SIMULATIONS OF QUANTUM ALGORITHMS USING THE QISKIT AER LIBRARY AND THE MATRIX PRODUCT STATE METHOD ON CPU AND GPU

Abstract: *This article presents a comprehensive study of the performance of quantum circuit simulation using the Matrix Product State (MPS) method within Qiskit Aer. The primary focus is on comparing the computational efficiency of implementations on central processing unit (CPU) and graphics processing unit (GPU) across various simulation platforms. Four key quantum algorithms are experimentally examined: the Variational Quantum Eigensolver (VQE), the Quantum Fourier Transform (QFT), the Quantum Approximate Optimization Algorithm (QAOA), and the Greenberger–Horne–Zeilinger (GHZ) state. For each algorithm, graphs are constructed showing the dependence of execution time on the number of qubits and the type of simulator. A single diagram is also provided to compare all four algorithms using the MPS method in Qiskit Aer. The results indicate that although the MPS simulator in Qiskit Aer is currently implemented only for CPU, it outperforms the traditional statevector simulator for low-entanglement circuits. At the same time, the lack of GPU acceleration limits the scalability of the MPS simulator. Analysis shows that implementing hardware acceleration of SVD operations using NVIDIA's cuQuantum and cuTensorNet libraries can provide a speedup of up to seven times. A comparative analysis of GPU implementations in Qiskit Aer and CUDA-Q is also conducted, including libraries such as cuStateVec, LAPACK, and cuSOLVER. Graphs are created to demonstrate the strengths and weaknesses of the simulators and simulation methods in Qiskit Aer. Notably, the MPS method remains one of the fastest by computation speed on CPU, but is sometimes surpassed by other methods on GPU. Further optimization directions for MPS are proposed, including the development of a hybrid CPU/GPU architecture and the utilization of modern GPU libraries to accelerate critical operations.*

Keywords: *quantum computing, quantum simulation, Qiskit Aer, MPS, statevector, cudaq.*

Постановка проблеми.

Розвиток квантових комп'ютерів відкриває нові можливості для розв'язання задач, що недоступні класичним системам. Однак сучасні квантові пристрої епохи NISQ (Noisy Intermediate-Scale Quantum)

обмежені кількістю кубітів і мають високий рівень шуму. У таких умовах симуляція квантових систем на класичних комп'ютерах залишається важливим інструментом для тестування квантових алгоритмів, дослідження потенціалу апаратного забезпечення та розробки методів виправлення помилок.

Класичне моделювання стикається з фундаментальною перешкодою — «прокляттям розмірності». Розмір гільбертового простору зростає експоненційно з кількістю кубітів, вимагаючи зберігання 2^N комплексних амплітуд для опису стану N-кубітної системи. Як наслідок, методи симуляції на основі повного вектора стану стають обчислювально нездійсненними вже при масштабах у кілька десятків кубітів, із практичною межею близько 50 кубітів навіть на найпотужніших суперкомп'ютерах. Це накладає серйозні обмеження на дослідження, оскільки фізичне обладнання швидко перевищує ці можливості.

Для подолання цього бар'єру було розроблено альтернативні підходи на основі тензорних мереж, серед яких найуспішнішим є представлення стану у формі матричного добутку станів (matrix product state, MPS).[1,2] Метод MPS розкладає один експоненційно великий тензор на ланцюжок менших тензорів, що дозволяє ефективно описувати квантові стани з низьким рівнем заплутаності, навіть для сотень кубітів. Ефективність MPS ґрунтується на «законі площі» для ентропії заплутаності і є теоретичною основою для таких потужних чисельних методів, як DMRG та TEBD.[3,4]

Ключовим, але й обчислювально найважчим кроком у симуляціях MPS є сингулярний розклад матриці (SVD). Після застосування двокубітного вентиля виконується SVD отриманого тензора з подальшим усіченням малих значень для контролю росту заплутаності. Цей крок, складність якого становить $O(\chi^3)$ (де χ — розмірність зв'язку), стає головним «вузьким місцем» симуляції. Таким чином, хоча MPS-формалізм обходить експоненційну залежність від кількості кубітів, його продуктивність напряму обмежується ефективністю виконання SVD.

Відкриті фреймворки, такі як Qiskit, стали ключовими інструментами для квантових досліджень. Його модуль Qiskit Aer[5] містить симулятор на основі MPS, що розширює коло доступних для аналізу задач. Однак тут виникає центральна проблема, яка є предметом цього дослідження: поточна реалізація симулятора MPS у Qiskit Aer виконана виключно для роботи на центральному процесорі (CPU) і відповідно SVD розклад також використовує лише ресурси CPU.

У сучасну епоху, коли графічні процесори (GPU) є стандартом для високопродуктивних наукових обчислень, це обмеження створює суттєвий розрив у можливостях. На відміну від CPU, оптимізованих для послідовних операцій, архітектура GPU з тисячами ядер і високою пропускною здатністю пам'яті ідеально підходить для масивно-паралельних задач, характерних для маніпуляцій з тензорними мережами. Відсутність підтримки GPU у симуляторі MPS призводить до десинхронізації між прогресом апаратних засобів і можливостями програмного забезпечення. Це заважає дослідникам моделювати системи з вищим рівнем заплутаності або змушує витрачати надмірну кількість часу на симуляції.

З огляду на зростаючу складність квантових алгоритмів, питання оптимізації продуктивності симуляції стає все більш актуальним. Одним із напрямків підвищення продуктивності є використання графічних процесорів (GPU) та адаптування методів оптимізації, таких як SVD, до GPU-архітектури, які, завдяки своїй архітектурі, орієнтованій на паралельні обчислення, можуть значно прискорити певні типи розрахунків або використання, або використання аналогів SVD, такі як QR, rSVD, які мають високоефективну реалізацію на GPU.[6]

Аналіз останніх досліджень і публікацій.

Упродовж останнього десятиліття спостерігається активне зростання інтересу до симуляції квантових систем за допомогою тензорних мереж, зокрема MPS. Фундаментальні засади методу MPS, що використовує SVD для контролю заплутаності, були узагальнені в оглядовій праці Schollwöck.[1] Подальший [2] розвиток чисельних методів на основі MPS, таких як TEBD та DMRG, продемонстрував ефективність MPS при моделюванні одномірних квантових систем.

Сучасна екосистема квантового програмного забезпечення, зокрема Qiskit Aer [5], підтримує MPS-базований симулятор (matrix_product_state) для класичного моделювання квантових схем [6]. Проте, як зазначається в офіційній документації IBM Qiskit, реалізація цього симулятора повністю базується на CPU і не використовує апаратне прискорення GPU, навіть для критично важливих операцій SVD, які реалізуються через LAPACK, зокрема, підпрограму zgesvd) [6-8].

Водночас сучасні дослідження вказують на значний потенціал GPU-архітектури для пришвидшення обчислень у MPS-симуляції. Зокрема, NVIDIA у межах SDK cuQuantum представила бібліотеку cuTensorNet, яка забезпечує апаратно прискорене виконання SVD та інших тензорних операцій на GPU [9,10]. У технічному звіті [11] вказується на прискорення обчислень SVD у 5–10 разів при використанні cuTensorNet у порівнянні з CPU-реалізаціями. Важливо, що бібліотека cuTensorNet

підтримує такі функції, як cutensornetTensorSVD та cutensornetTensorGateSplit, що є критичними для реалізації симуляції на основі MPS.

Компанія NVIDIA активно розвиває інструменти квантового прискорення симуляції за допомогою GPU, нова бібліотека CUDA-Q з глибокою інтеграцією cuQuantum SDK з більш зручним інтерфейсом доступу, вже інтегрує MPS-симулятори з підтримкою GPU, демонструючи потенціал такого підходу для симуляції схем з великою кількістю кубітів.[12,13] Таким чином, хоча в Qiskit ця функціональність відсутня, загальна тенденція в галузі вказує на неминучість інтеграції GPU-прискорення для симуляторів на основі тензорних мереж.

Альтернативні GPU-реалізації SVD, зокрема на базі MAGMA [14] та cuSOLVER [15], також продемонстрували високу ефективність для обчислювально-інтенсивних задач лінійної алгебри. Наприклад, згідно з оцінками [11], швидкість GPU-прискореного SVD значно перевищує продуктивність оптимізованих CPU-бібліотек, особливо для великих тензорів, де розмір зв'язку $\chi = 4096$.

У роботі [16] завдяки підвищенню пропускної здатності PCIe та оптимізації запуску ядер паралельного обчислення продемонстровано, що GPU перевершують CPU навіть на матрицях помірного розміру. Зокрема, SVD розміром 500 × 500 на NVIDIA Tesla V100 виконується за 0,218 мс — більш ніж у 100 разів швидше, ніж 25,14 мс, необхідних Intel MKL на Core i9-10920X.

Попри ці здобутки, на сьогодні відсутні публікації, що детально описують інтеграцію GPU-реалізованого MPS-симулятора безпосередньо в екосистему Qiskit. Це вказує на науково-технічну прогалину між теоретично обґрунтованими підходами до GPU-прискорення тензорних мереж і їх реалізацією в популярних квантових фреймворках. Крім того, недостатньо досліджено можливості реалізації таких симуляторів на споживчому апаратному забезпеченні (наприклад, NVIDIA RTX 4060 Ti), що обмежує застосування у широкій академічній спільноті.

Таким чином, існує обґрунтована потреба у практичному проектуванні та теоретичній оцінці GPU-прискореної симуляції MPS у Qiskit Aer з метою усунення існуючих архітектурних обмежень і підвищення ефективності симуляції.

Мета і задачі дослідження.

Метою даного дослідження є аналіз продуктивності симулятора на основі MPS у бібліотеці Qiskit Aer при використанні CPU, а також оцінка потенціалу його прискорення за допомогою архітектури GPU. Особлива увага приділяється виявленню обмежень у реалізації SVD на базі LAPACK (zgesvd) та дослідженню можливостей GPU-прискорення через сучасні інструменти, зокрема NVIDIA cuQuantum SDK. Для досягнення поставленої мети були визначені наступні задачі:

1. Охарактеризувати продуктивність існуючого симулятора Qiskit Aer MPS на CPU на основі аналізу його архітектури та відомих вузьких місць.
2. Дослідити теоретичні можливості та переваги використання GPU для прискорення обчислень в MPS, зокрема для операції сингулярного розклад матриці.
3. Сформулювати висновки щодо поточного стану симулятора MPS в Qiskit Aer та визначити перспективні напрямки для подальших досліджень з метою підвищення його продуктивності.
4. Порівняти симуляцію класу квантових алгоритмів на Qiskit Aer (CPU/GPU) та cudaq (GPU). Для цього було обрано обмежений список квантових алгоритмів QFT, GHZ, QAOA, VQE на основі статті [17] – основні характеристики цих алгоритмів наведено у Таблиці 1.

Таблиця 1

Таблиця використаних квантових алгоритмів та опис конфігурації квантових вентилів

Назва алгоритму	Опис	Кількість квантових вентилів	Кількість багато-кубітних квантових вентилів
QAOA	Quantum Approximate Optimisation Algorithm	$\frac{3}{2}N(N-1)+2N$	$N(N-1)$
QFT	Quantum Fourier transform	$\frac{1}{2}(N+1)+\frac{N}{2}$	$\frac{1}{2}(N^2-N)+\frac{N}{2}$
VQE	Variational Quantum Eigensolver	$6N-1$	$N-1$
GHZ	Greenberger-Horne-Zeilinger	$N-1$	$N-1$

1. QFT (квантове перетворення Фур'є) — квантовий аналог дискретного Фур'є-перетворення, реалізується H-гейтами та SWAP-ґейтами. [18]
2. GHZ — максимально заплутаний N-кубітний стабілізаторний стан; формується послідовністю H-ґейту та $N-1$ CNOT-ґейтів.[19]

3. QAOA — гібридний квантово-класичний алгоритм для мінімізації QUBO-задач; чергує виконання параметризованого Ansatz-ігрового кола й класичного оптимізатора, стійкий до шумів NISQ-пристроїв.[20]

4. VQE — гібридний метод для наближеного обчислення основного стану гамільтоніана; використовує параметризований Ansatz і класичну оптимізацію мінімальної енергії.[21]

В Таблиця 2 наведено список використаних симуляторів для оцінки Qiskit Aer MPS використаних для експериментального дослідження часу виконання

Таблиця 2

Таблиця з описом основних характеристик використаних квантових симуляторів та методів симуляції

Симулятор	Метод	GPU	CPU	Бібліотека для обчислення
Qiskit Aer	MPS	Ні	Так	LAPACK
Qiskit Aer	Statevector	Так	Так	cuStatevec
Nvidia Cudaq	Statevector	Так	Ні	cuStateVec

Методика експерименту

Експериментальна оцінка продуктивності симуляторів виконувалась у стандартизованих умовах із багатократними вимірюваннями часу на основі попередньої warm-up фази та стабілізації ресурсів системи. Для кожного квантового алгоритму (Таблиця 1) було здійснено 20 запусків для кожного типу симулятора (Таблиця 2) з фіксованим випадковим генератором і паузами між ітераціями для уникнення сторонніх впливів. Отримані дані оброблялись статистично: обчислювались середнє значення, стандартне відхилення, коефіцієнт варіації, довірчий інтервал. Усі вимірювання візуалізовано на графіках з відповідними позначеннями похибок.

Вимірювання часу симуляції проводилося на системі з обмеженою пам’яттю та обчислювальною потужністю. Цієї системи достатньо для дослідників, які порівняється метод MPS, а він є більш невибагливим до обчислювальної потужності та пам’яті.

- Операційна система: Ubuntu 2404 (Windows 11 5.15.167.4-microsoft-standard-WSL2)
- Відеокарта GPU NVIDIA 4060Ti 8GB
- Intel i5 12400F – 12 cores – 32GB RAM
- CUDA 12.9
- Qiskit 1.4.3
- Qiskit Aer 0.17
- Qiskit Aer gpu 0.14

Результати дослідження.

Симуляція на основі MPS в Qiskit Aer є особливо ефективним для квантових схем, де рівень заплутаності між кубітами є відносно низьким. На відміну від симулятора statevector, який зберігає повний вектор стану, симулятор MPS представляє квантовий стан у вигляді мережі тензорів. Продуктивність симуляції MPS сильно залежить від росту так званої "розмірності зв'язку", яка корелює з рівнем заплутаності в системі та кількістю багато-кубітних квантових вентилів. Ці характеристики запропонованих алгоритмів описані у Таблиці 2.

Результати візуалізовано на п’яти графіках (Рис. 1–5), які демонструють залежність часу виконання від кількості кубітів для різних алгоритмів і симуляторів. Графіки включають повні вимірювання з похибками та узагальнюють динаміку масштабованості обчислень. Нижче наведено детальний аналіз кожного випадку.

На (Рис. 1) QAOA MPS-симулятор на CPU демонструє найкращу стабільність при зростанні кількості кубітів до 25. Це пов’язано з особливістю QAOA-схем — малою кількістю двокубітних вентилів і низькою ентропією заплутаності. GPU-реалізації statevector (як у Qiskit, так і у CUDA-Q) показують експоненційне зростання після 18 кубітів, хоча до цього моменту GPU забезпечує прискорення до 3× порівняно з CPU. Рівень варіативності результатів GPU-графіка зростає після 20 кубітів, що може бути пов’язано з перевищенням кешу або обмеженнями пам’яті GPU.

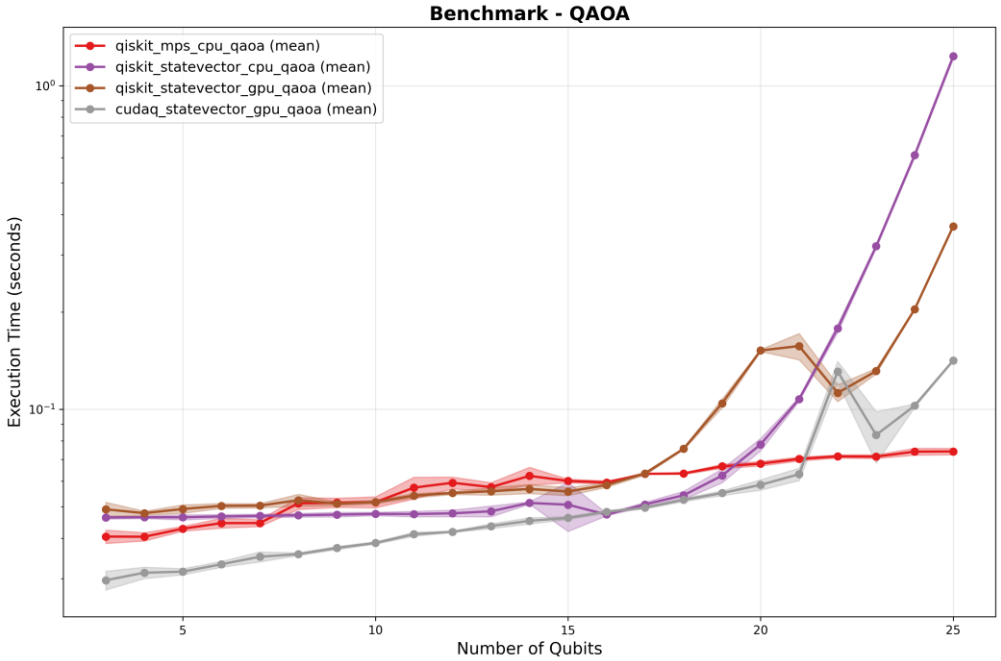

Рис. 1. Графік залежності часу виконання симуляції для алгоритму QAOA

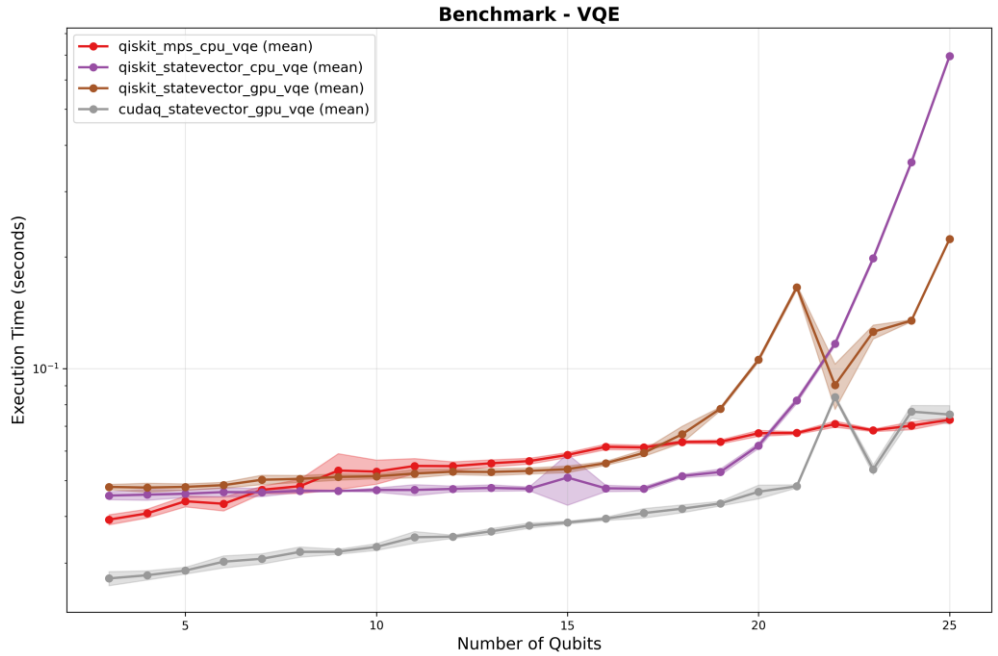

Рис. 2. Графік залежності часу виконання симуляції для алгоритму VQE

Алгоритм VQE (Рис. 2) має більшу глибину та ентропію, але демонструють схожі тенденції з QAOA. MPS на CPU залишається стабільним до 20 кубітів, після чого час симуляції зростає. GPU-графіки мають поступовий ріст і демонструють більшу варіативність. CUDA-Q стабільно демонструє найкращі результати в усьому діапазоні, особливо у високому діапазоні кубітів (22–25), де час виконання є нижчим у 2–2.5 рази від решти реалізацій.

QFT (Рис. 3) є найбільш вимогливим алгоритмом через високий рівень заплутаності та велику кількість двокубітних вентилів. MPS стає неефективним уже після 14–15 кубітів, що підтверджується різким зростанням часу симуляції. Statevector на GPU у CUDA-Q демонструє найкращу масштабованість і зберігає майже лінійне зростання часу до 25 кубітів.

Найпростіший алгоритм за структурою (GHZ) — лише один рівень двокубітних вентилів. Усі симулятори показують добру продуктивність, проте MPS на CPU має найнижчий час. Це підтверджує доцільність використання MPS для схем із передбачуваною топологією та лінійною заплутаністю.

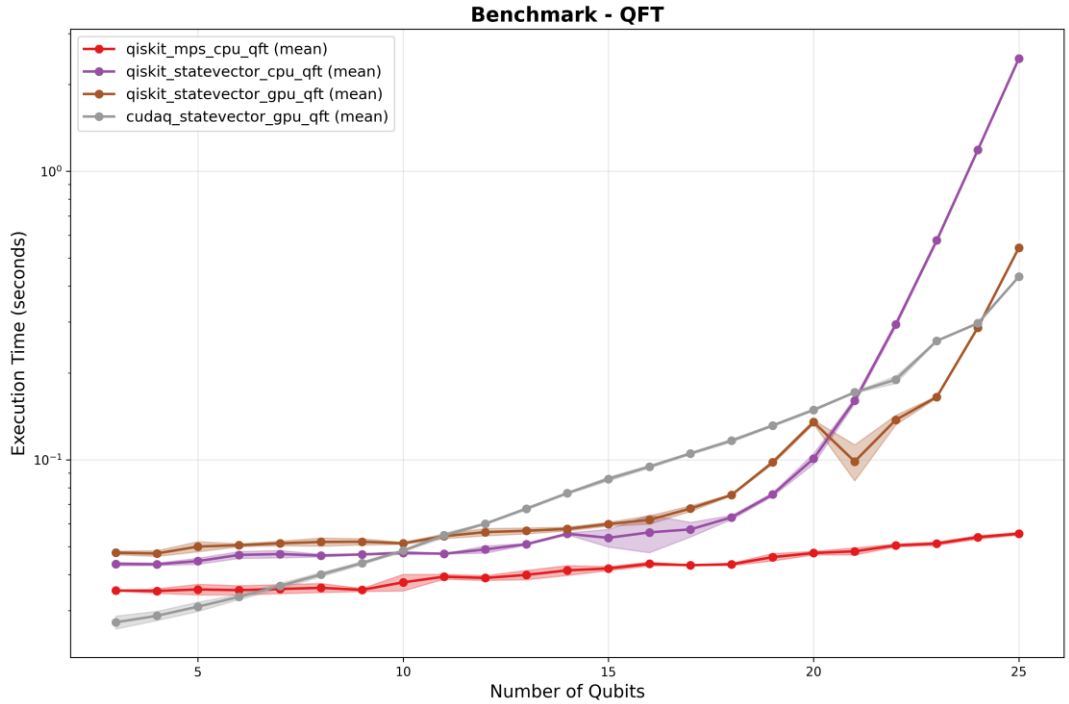

Рис. *3*. Графік залежності часу виконання симуляції для алгоритму QFT

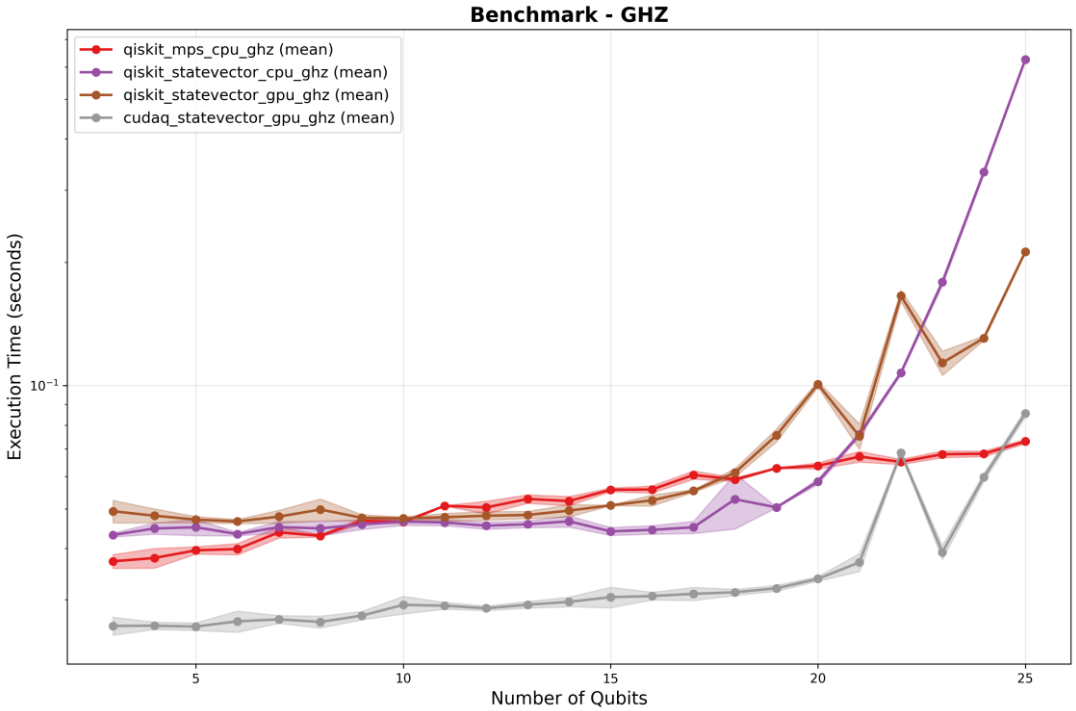

Рис. 4. Графік залежності часу виконання симуляції для алгоритму GHZ

Аналіз продуктивності на CPU показує, що симулятор MPS демонструє значно кращу продуктивність у порівнянні з симулятором statevector для схем з низькою заплутаністю, дозволяючи симулювати схеми з більшою кількістю кубітів за прийнятний час. Однак, зі збільшенням рівня заплутаності, продуктивність симуляції MPS суттєво знижується через зростання розмірності зв'язку, що робить SVD-розклад дедалі дорожчим.

Щодо потенційного використання GPU, варто зазначити, що хоча в Qiskit Aer прямої підтримки MPS немає, існують альтернативні шляхи. Один із них — застосування бібліотек, наприклад NVIDIA cuQuantum, яка пропонує оптимізовані для GPU функції для роботи з тензорними мережами. Інтеграція таких бібліотек із Qiskit у майбутньому може суттєво прискорити симуляцію MPS. Дослідження показують, що GPU можуть значно підвищити продуктивність для обчислень з тензорними мережами — основою MPS — особливо при роботі з великими матрицями, що виникають при високій розмірності зв’язків.[9,11]

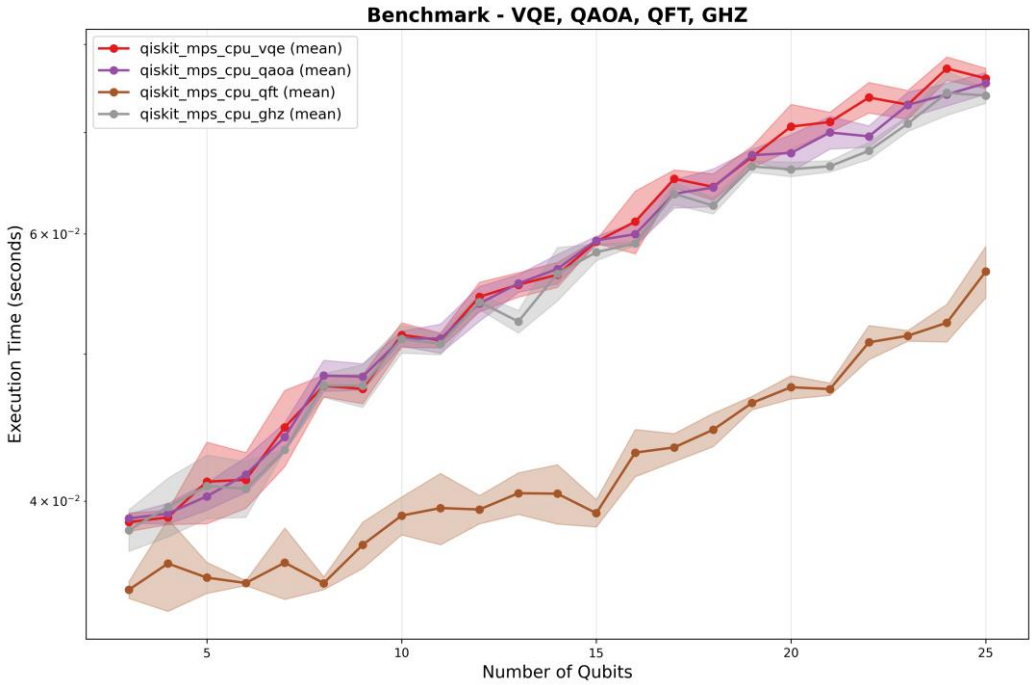

Рис. 5. Графік залежності часу виконання MPS симуляції за допомогою бібліотеки Qiskit Aer для алгоритмів VQE, QAOA, QFT, GHZ

Висновки і перспективи подальших досліджень

Проведене дослідження дозволяє зробити наступні висновки:

1. Під час дослідження на мові програмування Python створено бенчмарк [20] для оцінки швидкості симуляція чотирьох алгоритмів використовуючи чотири симулятора, а також досліджено вплив використання CPU/GPU на швидкість симуляції.
2. Метод симуляції на основі MPS у бібліотеці Qiskit є потужним інструментом для аналізу квантових схем із великою кількістю кубітів і низьким рівнем заплутаності, але його продуктивність обмежена виконанням на CPU і в деяких випадках програє CUDAQ Statevector, навіть для алгоритмів із обмеженою заплутаністю та малою кількістю кубітів.
3. На даний момент симулятор MPS у Qiskit Aer не має прямої підтримки GPU, що є значним недоліком порівняно з іншими методами симуляції в тій же бібліотеці та конкуруючими фреймворками.[6,12]
4. Основним обчислювальним вузьким місцем у симуляції MPS є операція розкладу SVD, що виконується багаторазово для контролю росту заплутаності. Тому дослідження використання різних імплементацій цього алгоритму матиме суттєве значення.

Перспективи подальших досліджень пов'язані з інтеграцією GPU-прискорення та відповідних бібліотек у симулятор Qiskit Aer MPS. Ключовим напрямком є оцінка ефективності реалізації SVD на GPU в контексті MPS. Спеціалізовані бібліотеки, такі як NVIDIA cuQuantum, пропонують оптимізовані функції для SVD, які можуть забезпечити значне прискорення порівняно з реалізаціями на CPU з допустимою точністю. Майбутні дослідження повинні бути зосереджені на:

- Розробці та тестуванні прототипу симулятора MPS у Qiskit, який використовує GPU-бібліотеки для виконання SVD та інших тензорних операцій.
- Порівняльний аналіз продуктивності MPS на CPU та прототипів реалізації на GPU для визначення оптимального підходу з точки зору швидкості та використання пам'яті для різних розмірностей зв'язку.
- Дослідження гібридних підходів, де частина обчислень виконується на CPU, а обчислювально-інтенсивні кроки, як-от SVD, переносяться на GPU, щоб мінімізувати накладні витрати на передачу даних.
- Порівняння точності квантових симуляцій прототипів бібліотеки Qiskit Aer MPS на GPU, оскільки MPS є методом з більшою похибкою в порівнянні із Statevector. Для реальних задач важливо контролювати цю похибку в заданих межах і розуміти її природу та величину.

Подальші дослідження у цій галузі допоможуть значно збільшити швидкість симуляції MPS у Qiskit Aer, зробивши її більш ефективним інструментом для ширшого набору квантових задач.

Список використаної літератури

1. Schollwöck U. The density-matrix renormalization group in the age of matrix product states / Ulrich Schollwöck // Annals of Physics. – 2011. – Vol. 326, no. 1. – P. 96–192. https://doi.org/10.1016/j.aop.2010.09.012
2. Orús R. A practical introduction to tensor networks: matrix product states and projected entangled pair states // Annals of physics. – 2014. – Vol. 349. – P. 117–158. https://doi.org/10.1016/j.aop.2014.06.013
3. Bacchetta A. P. Methods of Quantum Circuit Simulation: A Comprehensive Comparative Analysis : Master Thesis / Bacchetta Anthony Peter. – [S. l.], 2025. https://repository.rit.edu/cgi/viewcontent.cgi?article=13309&context=theses
4. Combining matrix product states and noisy quantum computers for quantum simulation / Baptiste Anselme Martin [et al.] // Physical review A. – 2024. – Vol. 109, no. 6. https://doi.org/10.1103/physreva.109.062437
5. Aer - high performance quantum circuit simulation for Qiskit // GitHub https://github.com/Qiskit/qiskit-aer
6. AerSimulator (v0.26). *IBM Quantum Documentation*. URL: https://quantum.cloud.ibm.com/docs/api/qiskit/0.26/qiskit.providers.aer.AerSimulator
7. Quantum computer simulations at warp speed: assessing the impact of GPU acceleration: A case study with IBM qiskit aer, nvidia thrust & cuquantum / J. Faj et al. *2023 IEEE 19th International Conference on e-Science (e-Science)*, Limassol, Cyprus, 9–13 October 2023. 2023. https://doi.org/10.1109/e-science58273.2023.10254803
8. LAPACK – linear algebra package. *https://netlib.org/lapack/*
9. NVIDIA cuQuantum SDK. https://developer.nvidia.com/cuquantum-sdk
10. Gao Y., Lubowe T. Enabling matrix product state–based quantum circuit simulation with NVIDIA cuquantum. *https://developer.nvidia.com/blog/enabling-matrix-product-state-based-quantum-circuit-simulation-with-nvidia-cuquantum/*
11. Cuquantum sdk: a high-performance library for accelerating quantum science / H. Bayraktar et al. *2023 IEEE international conference on quantum computing and engineering (QCE)*. 2023. P. 1050–1061. https://ieeexplore.ieee.org/document/10313722
12. Schieffer G., Markidis S., Peng I. Harnessing cuda-q's MPS for tensor network simulations of large-scale quantum circuits. *2025 33rd euromicro international conference on parallel, distributed, and network-based processing (PDP)*, Turin, Italy, 12–14 March 2025. 2025. P. 94–103. https://doi.org/10.1109/pdp66500.2025.00022
13. CUDA Quantum Simulation Backends. *https://nvidia.github.io/cuda-quantum/0.6.0/using/simulators.html*
14. Dongarra J. J., Tomov S. Matrix Algebra for GPU and Multicore Architectures (MAGMA) for Large Petascale Systems. Office of Scientific and Technical Information (OSTI), 2014. https://doi.org/10.2172/1126489
15. CuSOLVER API reference. *https://docs.nvidia.com/cuda/cusolver*.
16. Efficient GPU implementation of randomized SVD and its applications / Ł. Struski et al. *Expert systems with applications*. 2024. Vol. 248. P. 123462. https://doi.org/10.1016/j.eswa.2024.123462
17. Vallero M., Rech P., Vella F. State of practice: Evaluating GPU performance of state vector and tensor network methods. *Future generation computer systems*. 2026. Vol. 174. P. 107927. https://doi.org/10.1016/j.future.2025.107927
18. Coppersmith D. An approximate Fourier transform useful in quantum factoring. *ArXiv preprint*. quant-ph/0201067. https://doi.org/10.48550/arXiv.quant-ph/0201067
19. Greenberger D. M., Horne M. A., Zeilinger A. Going beyond bell's theorem. *Bell's theorem, quantum theory and conceptions of the universe*. Dordrecht, 1989. P. 69–72. https://doi.org/10.1007/978-94-017-0849-4_10
20. Qiskit Aer MPS evaluation experiment exploring // Github, https://github.com/DimitrYo/mps_experiment
21. Y. Kostiuk, et al., Integrated protection strategies and adaptive resource distribution for secure video streaming over a Bluetooth network, in: Cybersecurity Providing in Information and Telecommunication Systems II, vol. 3826 (2024) 129–138.
22. S. Zhebka, et al., Method for Adaptive Allocation of Cryptographic Resources in Distributed Databases, in: Cybersecurity Providing in Information and Telecommunication Systems, vol. 3991 (2025) 620–628.