Інструменти Квантових Обчислень. Мова Програмування Q#

Іван Божко
кафедра програмної інженерії
Харківський національний університет
радіоелектроніки
Харків, Україна
ivan.bozko@nure.ua

Григорій Четвериков
кафедра програмної інженерії
Харківський національний університет
радіоелектроніки
Харків, Україна
grigorij.chetverykov@nure.ua

Quantum Computing Tools. The Q# Programming Language

Ivan Bozhko
Department of Software Engineering
Kharkiv National University
of Radio Electronics
Kharkiv, Ukraine
ivan.bozhko@nure.ua

Grygoriy Chetverykov
Department of Software Engineering
Kharkiv National University
of Radio Electronics
Kharkiv, Ukraine
grigorij.chetverykov@nure.ua

***Анотація*—У даній роботі розглядається сучасний стан існуючих інструментів для квантових обчислень, особлива увага приділяється мові програмування Q# як основного інструменту на сьогоднішній день.**

***Abstract*—This work deals with the current state of the existing tools for quantum computing, especially the Q# programming language as the main tool for this nowadays.**

Ключові слова—квантові обчислення, квантовий комп'ютер, Q#, інструменти квантових обчислень.

Keywords—quantum computing, quantum computer, Q#, quantum computing tools.

I. Вступ

Квантові обчислення являють собою альтернативний підхід до автоматизованих обчислень за допомогою квантових комп'ютерів.

Незважаючи на розвиток досліджень у цій галузі сьогодні, вперше ідею квантових обчислень висловив радянський математик Ю.І. Манін ще у 1980 році в своїй відомій монографії «Обчисленне та необчисленне» [1]. Однак більший інтерес до даного виду обчислень виник лише у 1982 році після того, як американським фізиком-теоретиком Річардом Фейнманом було зауважено, що не всі квантово-механічні операції можна в точності переносити на класичний комп'ютер і більш ефективно здійснювати їх за допомогою квантових операцій.

Додаткової актуальності проблемі дослідження квантових обчислень додав математик Пітер Шор, який у 1994 році запропонував алгоритм, що дозволяє розкладати n-значне число на прості множники з поліноміальною складністю. На класичних комп'ютерах ця задача носить набагато більшу складність і не дозволяє отримати результат за задовільний час.

Оскільки дана задача є основою багатьох популярних криптографічних алгоритмів (наприклад, RSA) [2], можлива поява квантових комп'ютерів ставить під загрозу безпеку при обміні даними в мережі і після появи реального прототипу квантового комп'ютеру може стати загальносвітовою проблемою безпеки.

Отже, квантові обчислення є актуальними не тільки для наукових проблем моделювання квантових процесів, а також є актуальними для світу інформаційних технологій. З метою їх популяризації не тільки в науковому світі, але і серед розробників, створюються інструменти для спрощення роботи з квантовими алгоритмами, такі як мова програмування Q#, що розглядається в даній роботі.

II. Існуючі інструменти квантових обчислень

За останні кілька років з ростом популярності досліджень з квантових обчислень почали з'являтися інструменти та емулятори квантових комп'ютерів, що дозволяють спробувати обчислення на практиці. Наведемо декілька популярних інструментів з описом.

A. *Microsoft Quantum Development Kit*

Корпорація Microsoft випустила доступний для попереднього перегляду Quantum Development Kit, який містить нову квантову мову програмування Q#, інтеграцію з середовищем розробки Visual Studio, симулятори, які працюють як з локальною системою, так і з їх потужною хмарною платформою Azure, а також бібліотеками та зразками коду, які можуть використовуватися як конструктивні блоки.

B. *IBM Quantum Experience*

IBM створила експериментальний квантовий процесор на 5 кубітів, що доступний користувачам через мережу Інтернет. На веб-сайті IBM Quantum Experience можна знайти короткий підручник, який пояснює основи квантових обчислень та інструкції щодо використання системи, налаштування інтерфейсів доступу до кубітів, симулятор, який дозволяє імітувати їх конфігурацію перед тим, як запустити її на фактичній машині, і доступ до самої машини, яка дозволяє запускати конфігурацію та переглядати результати.

C. *Rigetti Forest*

Пакет Rigetti Forest складається з квантової мови інструкцій під назвою Quil, відкритої бібліотеки Python для побудови програм Quil під назвою pyQuil, бібліотеки квантових програм під назвою Grove і середовища моделювання, що називається QVM (Quantum Virtual Machine). pyQuil і Grove – це програми з відкритим кодом, доступні на Github. Користувачі можуть розробляти свої програми, використовуючи pyQuil і Grove на власному комп'ютері, а потім передавати їх QVM для моделювання через веб-портал, який доступний для зареєстрованих користувачів.

D. *ProjectQ*

ProjectQ - це відкрите програмне забезпечення для квантових обчислень, реалізоване мовою Python. Це дозволяє користувачам реалізувати свої квантові програми мовою Python за допомогою потужного та інтуїтивного синтаксису. ProjectQ може потім транслювати ці програми на будь-яку серверну частину: симулятор, що працює на класичному комп'ютері, або квантовий комп'ютер (наприклад, з використанням IBM Quantum Experience). Інші апаратні платформи на даний момент не підтримуються.

Окрім зазначених інструментів, існують інші, такі як Cirq, Quirk, QuTiP, але вони є менш потужними ніж описані вище.

Як видно з опису існуючих рішень, більшість з них є доповненням до існуючих мов програмування (зокрема Python), однак враховуючи нову парадигму обчислень це може викликати складнощі при програмуванні алгоритмів за допомогою цих інструментів, тому більш гнучким та потужним рішенням є окрема мова програмування, яку на даний момент пропонує лише Microsoft.

Окремо необхідно виділити рішення від Intel з існуючим експериментальним квантовим комп'ютером, але воно вимагає виконання запитів до окремих кубітів через API, що ускладнює його використання.

III. Мова Q#

Як було розглянуто вище, складовою частиною Microsoft Quantum Development Kit є мова Q#, спеціально розроблена для квантових обчислень. З точки зору програмної інженерії, таке рішення є найбільш цікавим, оскільки дозволяє абстрагуватися від парадигми класичних обчислень і класичних мов програмування та описувати квантовий алгоритм за допомогою спеціального синтаксису.

Розглянемо мову Q# детальніше.

A. *Модель обчислень*

Згідно офіціальній документації Microsoft [3], квантовий комп'ютер слід розглядати як співпроцесор, аналогічно до графічних процесорів. Основна логіка додатків виконується на класичному комп'ютері, що називають хостом, який, в свою чергу, може викликати квантову підпрограму. Після виконання роботи підпрограми управління повертається основній програмі та вона продовжує своє виконання.

Таким чином, в даній моделі є три рівні обчислень:

- класичні обчислення, що готують дані для квантової програми;
- власне квантові обчислення;
- класичні обчислення, що викликаються для потреб квантової програми.

Необхідності розробляти усі три рівні обчислень одною мовою немає. Більше того, враховуючи особливості контрольних структур при квантових обчисленнях, комбінування різних мов для класичних та квантових частин програми буде семантично правильнішим.

B. *Особливості мови*

Мова Q# забезпечує невеликий набір примітивних типів, а також два свособи (масиви та кортежі) для створення нових типів даних (що свідчить про більшу обмеженість мови в порівнянні з класичними з метою оптимізації) В цілому мова підтримує звичайно процедурну модель програмування з умовними конструкціями (if-then) та циклами.

Підтримуються основні типи даних (Int, Double, Bool, String), однак додано кілька особливих типів даних, пов'язаних з квантовою парадигмою обчислень.

Тип Qubit являє собою квантовий біт. Даний тип є інкапсульованим від користувача і єдиною операцією над ним є операція порівняння. Всі інші дії над ними реалізуються за допомогою методів стандартної бібліотеки.

Іншим особливим типом є тип Pauli – елемент одиничної групи Паулі. Використовується для базової операції обертання та для визначення осови вимірювання. Є дискримінованим об'єднанням з чотирма можливими значеннями: PauliI, PauliX, PauliY, PauliZ.

Також існує тип Result, що приймає значення One або Zero та інкапсулює результат вимірювання.

Особливої уваги заслуговує поняття функтора. Функтор у Q# – це фабрика, яка дозволяє визначати нові операції з інших. Вони мають доступ до базової операції при визначенні нової. Таким чином, вони можуть виконувати більш складні функції, ніж базові.

З перерахованих особливостей мови очевидно, що наявність особливих типів даних та мовних конструкцій дозволяють більш зрозумілим чином описувати квантові алгоритми без додаткового коду, який було б необхідно використовувати при написанні коду з використанням бібліотеки та класичної мови програмування.

Наприклад, в Q# виділення кубіту організовано подібно до мови C# за допомогою оператора using, що виглядає значно простіше, ніж використання кубітів через API.

C. Приклади написання програм

З метою наочної ілюстрації можливостей мови квантових обчислень Q# створимо просту програму в серидовищі Visual Studio. Для цього треба мати встановленим Microsoft Quantum Development Kit та створити додаток типу «Q# Application». Після цього серидовище буде готовим до написання коду.

Проілюструємо просту програму встановлення стану кубіту (рис. 1).

```
open Microsoft.Quantum.Primitive;
open Microsoft.Quantum.Canon;

operation Set (desired: Result, q1: Qubit) : ()
{
    body
    {
        let current = M(q1);
        if (desired != current)
        {
            X(q1);
        }
    }
}
```

Рис. 1. Програма встановлення стану кубіту

З даного коду можна побачити, що стандартні інструменти дозволяють виконувати базові операції. Таким чином, розробник підключає необхідні інструменти за допомогою директиви open і використовує їх в коді.

В логіці даної програми створюється функція Set, яка приймає параметри типу Result та Qubit, а далі в тілі функції отримується стан кубіту за допомогою стандартної функції M і, якщо результат не є таким, який потрібно встановити, викликається стандартна функція обернення стану кубіту X.

З даної програми можна побачити деякі стандартні засоби роботи з кубітами, які відрізняють мову Q# від інших інструментів. Треба зазначити, що такий спосіб роботи з кубітами потребує значно менше додаткового коду, ніж, наприклад, робота з API при використанні IBM Quantum Experience.

IV. Висновки

В даній роботі було стисло розглянуто основні інструменти для квантових обчислень, а також описано актуальність такого дослідження на сьогоднішній день. Особливу увагу було приділено мові Q# як інструменту, що націлений на найбільш широку аудиторію користувачів (не тільки науковців, але й розробників) та пропонує широкі можливості стандартних засобів квантових обчислень, не вимагаючи від користувача реалізовувати основні операції самостійно.

Слід зазначити, що в даній роботі не оцінювалася ефективність виконання програм при використанні розглянутих інструментів, оскільки на даному етапі точність її оцінки була б низькою через відсутність стандартизованого та повністю робочого прототипу квантового комп’ютера, що, як наслідок, означає відмінність реалізації кожного окремого комп’ютера.

Очевидно, що із швидким розвитком даної галузі на сьогоднішній день будуть з’являтися нові інструменти і вдосконалюватися сучасні, що потребуватиме аналогічного дослідження через деякий час. Однак дана робота дає високорівневий опис можливостей мови Q#, яка, на думку авторів, є найбільш зручним способом практичного досліджень квантових обчислень на сьогодні.

Література References

[1] Ю. И. Манин. Вычислимое и невычислимое. / Манин И. Ю. –М.: Сов. радио, 1980. – 128 с.

[2] L. Chen. Report on Post-Quantum Cryptography. / Lily Chen, Stephen Jordan, Yi-Kai Liu, Dustin Moody, Rene Peralta, Ray Perlner, Daniel Smith-Tone – NIST Interagency Report 8105, 2016. – 15 p.

[3] The Q# Programming Language [Online]. Available: https://docs.microsoft.com/en-us/quantum/quantum-qr-intro?view=qsharp-preview