УДК 004.438
DOI 10.25205/1818-7900-2020-18-2-43-53

Инструменты для выполнения и эмуляции квантовых вычислений

П. Е. Баскаков, Ю. Ю. Хабовец, И. А. Пилипенко
В. О. Кравченко, Л. В. Черкесова

Донской государственный технический университет
Ростов-на-Дону, Россия

Аннотация

В настоящее время квантовые технологии находятся на передовой развития научной мысли. Крупные корпорации создают собственные квантовые суперкомпьютеры, разрабатываются квантовые аналоги классических алгоритмов, ведутся исследования в области квантовой криптографии. Но так как квантовые компьютеры еще не получили широкого распространения, актуальным становится вопрос: как обычным пользователям, ученым и исследователям не отставать от развития науки? Одним из возможных решений является использование различного рода инструментов для эмуляции квантовых вычислений на локальном неквантовом компьютере. Кроме того, существует также возможность получить в распоряжение несколько кубит квантового суперкомпьютера IBM. Как правило, такие инструменты реализуются в виде библиотек для различных языков программирования. Ввиду того что работа с реальными квантовыми компьютерами доступна лишь узкому кругу исследователей, эмуляторы просто необходимы для проверки гипотез или алгоритмов. В данной статье рассмотрены наиболее популярные квантовые эмуляторы, использующиеся для квантовых вычислений и позволяющие эмулировать процесс работы квантового компьютера. Были изучены квантовые эмуляторы, выявлены и описаны их индивидуальные особенности, составлены рекомендации для более удобного начала работы с ними, а также описаны их достоинства и недостатки. Произведен обзор нескольких библиотек для языков JavaScript, Python, C/C++, а также рассмотрены средство с веб-интерфейсом (Quantum Programming Studio) и набор инструментов от компании Microsoft (Microsoft Quantum Development Kit), основным языком которого служит Q#.

Ключевые слова

квантовые вычисления, библиотеки, эмуляция, кубит, языки программирования

Для цитирования

Баскаков П. Е., Хабовец Ю. Ю., Пилипенко И. А., Кравченко В. О., Черкесова Л. В. Инструменты для выполнения и эмуляции квантовых вычислений // Вестник НГУ. Серия: Информационные технологии. 2020. Т. 18, № 2. С. 43–53. DOI 10.25205/1818-7900-2020-18-2-43-53

Tools for Performing and Emulating Quantum Computing

P. E. Baskakov, Yu. Yu. Khabovets, I. A. Pilipenko
V. O. Kravchenko, L. V. Cherkesova

Don State Technical University
Rostov on Don, Russian Federation

Abstract

Currently, quantum technologies are at the forefront of scientific thought. Large corporations are creating their own quantum supercomputers, developing quantum analogues of classical algorithms, and research is being conducted in the field of quantum cryptography. But since quantum computers have not yet become widespread, the question be-

comes relevant: how can ordinary users, scientists and researchers keep up with the development of science? One possible solution is to use various kinds of tools to emulate quantum computing on a local non-quantum computer. In addition, there is also the opportunity to have several qubits of IBM's quantum supercomputer available. As a rule, such tools are implemented in the form of libraries for various programming languages. Due to the fact that working with real quantum computers is available only to a narrow circle of researchers, emulators are simply necessary to test hypotheses or algorithms. This article examined the most popular quantum emulators used for quantum computing and allowing emulating the process of a quantum computer. Work was carried out to study quantum emulators, to identify and describe their individual characteristics, to make recommendations for a more convenient start to work with them, as well as to describe their advantages and disadvantages. A review of several libraries for the JavaScript, Python, C / C ++ languages was made, as well as a tool with a web interface (Quantum Programming Studio) and a set of tools from Microsoft (Microsoft Quantum Development Kit), the main language of which is Q #, is examined. At the end of the article, a conclusion is made regarding the considered tools.

Keywords

quantum computations, libraries, emulation, qubit, programming languages

For citation

Baskakov P. E., Khabovets Yu. Yu., Pilipenko I. A., Kravchenko V. O., Cherkesova L. V. Tools for Performing and Emulating Quantum Computing. *Vestnik NSU. Series: Information Technologies*, 2020, vol. 18, no. 2, p. 43–53. (in Russ.) DOI 10.25205/1818-7900-2020-18-2-43-53

Введение

Идея применения квантовых механизмов для создания устройств, названных впоследствии квантовыми компьютерами, была подана советским ученым Юрием Маниным в работе «Вычислимое и невычислимое» [1], позднее в работе «Simulating physics with computers» Ричард Фейнман продолжил эти рассуждения [2]. Квантовый компьютер является единственной на сегодняшний день моделью, способной предложить экспоненциальный прирост скорости вычислений по сравнению с обычными современными компьютерами, пусть это и касается лишь ограниченного круга задач [3].

В статье «Quantum supremacy using a programmable superconducting processor» вводится такое понятие, как «квантовое превосходство». Это потенциальная способность квантовых вычислительных устройств решать проблемы, которые классические компьютеры практически не могут, или им требуется для этого слишком много времени [4]. В той же статье был дан критерий наступления квантового превосходства: по мнению авторов, это порог вычислительной мощности в 50 кубитов. Таким образом, квантовое превосходство было достигнуто, а недавно созданная 72-кубитная квантовая машина Google Bristlecone уже превосходит классический суперкомпьютер по четко определенной вычислительной задаче. Но, к сожалению, таких компьютеров в мире всего несколько, поэтому большинству исследователей еще долго придется довольствоваться эмуляторами квантовых вычислений.

Целью данной работы является обзор современных инструментов для работы с квантовыми вычислениями и компьютерами. В данной статье можно ознакомиться с существующим инструментарием разработчика квантовых приложений, прикоснуться к реальным квантовым вычислениям и выбрать для себя наиболее подходящий инструмент для работы с квантовым эмулятором или компьютером.

1. **Quantum circuit** – это библиотека с открытым исходным кодом, предназначенная для эмуляции квантовых вычислений в программах на языке программирования JavaScript. Благодаря этому можно его использовать прямо в браузере при создании html-страницы либо на сервере node.js. Также можно применять quantum cirquit внутри Jupyter Notebook, однако в этом случае понадобится установка дополнительного плагина. Созданные квантовые схемы доступны для импорта / экспорта в форматы других распространенных эмуляторов (Qiskit, Cirq, QuEST и др.) и для сохранения в виде векторных рисунков.

По заверениям разработчиков библиотеки, quantum cirquit способен обеспечивать эмуляцию более 20 кубит без существенного влияния на производительность системы [1]. Для этого используется ряд «ухищрений», призванных оптимизировать потребление оперативной памяти. В частности, алгоритм симуляции не хранит полный вектор состояний в памяти в виде массива размера 2^n, где n – количество кубит в схеме, а использует специальную структуру данных, в которой сохранены только ненулевые значения. Элементы матрицы преобразований рассчитываются, умножаются на вектор состояний и сохраняются «на лету», таким образом позволяя достичь единовременного хранения максимум 2 векторов состояний. Алгоритм допускает распараллеливание, то есть проведение расчетов возможно в том числе с использованием графического ускорителя (Graphics Processing Unit, GPU), однако данная особенность в настоящее время не реализована.

Рассмотрим теперь пример работы с библиотекой на основе следующего листинга кода на языке JavaScript:

```
var circuit = new QuantumCircuit(2);
circuit.addGate("h", 0, 1);
circuit.addMeasure(1, "c", 0);
circuit.run();
console.log(circuit.getCregValue("c")).
```

В данном примере создается схема с двумя кубитами, ко второму из которых применяется преобразование Адамара с последующим измерением кубита и сохранением результата в 0-й бит классического не квантового регистра *c*, после чего схема запускается на выполнение и на экран выводится значение регистра. Следует отметить, что нужные регистры будут добавлены на схему автоматически, если ранее этого не было сделано явным образом. Полный список реализованных преобразований доступен на странице документации библиотеки [2].

Для определения сравнительных признаков рациональным будет выделить достоинства и недостатки пакета quantum–circuit.

К достоинствам можно отнести:

- открытость исходного кода;
- хорошую документированность;
- возможность интеграции с форматами других библиотек и оптимизированное потребление памяти.

Главным недостатком является несбалансированное использование ресурсов вычислительной системы [3]. Так, реализация проведения вычислений на GPU позволила бы снять часть нагрузки с процессора и сократить время, требуемое на обработку.

2. **QuEST.** Вышеописанный недостаток устранен в другой популярной библиотеке квантовых вычислений – QuEST (Quantum Exact Simulation Toolkit), разрабатываемой специалистами исследовательской группы Оксфордского университета QTechTheory.

Данная библиотека предназначена для использования совместно с языками программирования C / C++. Благодаря этому достигается возможность использования ее на любой целевой платформе – от ноутбуков до суперкомпьютеров, требуется лишь наличие соответствующего компилятора.

Библиотека направлена на точную и эффективную симуляцию глубоких квантовых схем с большим числом кубитов. Отличительной особенностью является возможность построения распределенной сети, что позволяет объединить вычислительные мощности нескольких ком-

[1] Quantum Circuit Simulator. 2020. URL: https://www.npmjs.com/package/quantum-circuit (дата обращения 15.03.2020).

[2] Там же.

[3] List of QC simulators / Quantiki – Portal and Wiki, 2020. URL: https://quantiki.org/wiki/list-qc-simulators (дата обращения 12.04.2020).

пьютеров-узлов для достижения наибольшей производительности. Так, с использованием 2 048 компьютеров ARCUS и ARCHER удалось смоделировать 38 кубит [5].

Алгоритмы симуляции являются многопоточными, задействуются не только мощности центрального процессора, но и GPU, для чего требуется наличие библиотек Nvidia CUDA (Compute Unified Device Architecture). Сообщается, что с использованием видеоадаптера с объемом памяти 2 Гб возможно моделирование 26 кубитов, на ноутбуке с 16 Гб оперативной памяти – 29 кубитов [4].

Работа с библиотекой должна начинаться с вызова функции *createQuESTEnv*, которая вернет объект класса *QuESTEnv* – окружение, инкапсулирующее конфигурацию многопоточности, распределенности и GPU-ускорения.

Рассмотрим следующий код на языке программирования C++:

```
QuESTEnv env = createQuESTEnv();
Qureg qubits = createQureg(3, env);
initZeroState(qubits);
hadamard(qubits, 0);
controlledNot(qubits, 0, 1);
rotateY(qubits, 2, .1);
destroyQureg(qubits, env);
destroyQuESTEnv(env).
```

Квантовые регистры создаются вызовом функции *createQreg*. Затем регистры могут быть инициализированы нулевым состоянием (*initZeroState*) либо положительным (*initPlusState*).

Далее к регистрам можно применять преобразования (которые тут называются *гейтами*). Список реализованных гейтов доступен по ссылке [5].

При завершении работы рекомендуется освободить память, выделенную под регистры и окружение (функции *destroyQureg* и *destroyQuESTEnv* соответственно).

На основании вышесказанного можно сделать вывод, что главное преимущество QuEST заключается в комбинировании производительности центрального процессора (Central Processing Unit, CPU) и GPU, а также возможности проведения распределенных вычислений. На данный момент это единственный симулятор с открытым исходным кодом, обладающий подобным функционалом [5]. Также следует отметить подробную документированность и наличие примеров схем в официальном репозитории.

3. **Qiskit** – фреймворк для квантовых вычислений с открытым исходным кодом, разрабатываемый исследовательской группой IBM Research, а также сообществом энтузиастов с целью создания ПО для облачных квантовых вычислений.

Основная версия Quiskit использует Python в качестве языка программирования, однако доступны также версии для языков Swift и JavaScript [6]. Quiskit предоставляет возможность разработки квантового ПО как на высоком уровне абстракции (для пользователей без опыта квантового программирования), так и на низком уровне, близком к машинному коду OpenQASM. Такая возможность обеспечивается благодаря использованию 4-х компонентов: Terra, Aqua, Aer, Ignis.

- Qiskit Terra является своего рода ядром фреймворка [7]. Данный модуль предоставляет инструменты для создания квантовых схем на машинном или близком к нему уровне, а также квантовых гейтов. Кроме этого, Qiskit Terra содержит инструменты для оптимизации кванто-

[4] Quantum Exact Simulation Toolkit, 2020. URL: https://quest.qtechtheory.org/about/ (дата обращения 02.04.2020).

[5] QuEST Coding / Quantum Exact Simulation Toolkit, 2020. URL: https://quest.qtechtheory.org/docs/#coding (дата обращения 02.04.2020).

[6] Qiskit / Wikipedia – The free Encyclopedia, 2020. URL: https://en.wikipedia.org/wiki/Qiskit (дата обращения 10.04.2020).

[7] Welcome to Quantum / Quantum computing software, 2020. URL: https://qiskit.org/ (дата обращения 24.03.2020).

вых схем под конкретные типы вычислительных систем и версии бэкенда (локальный эмулятор или удаленные вычисления).

- Qiskit Aqua может быть использован без непосредственного квантового программирования. Данный модуль предоставляет набор инструментов для решения кросс-квантовых задач [6]. В настоящее время доступны эксперименты в области химии, искусственного интеллекта, оптимизации и финансовой сферы. Пользователь может определить какую-либо проблему и получить решение, а Qiskit Aqua реализует соответствующий квантовый алгоритм.
- Qiskit Aer предоставляет высокопроизводительный симулятор для всего стека технологий. Он содержит в себе оптимизированный C++ – симулятор для выполнения схем, скомпилированных в Qiskit Terra, а также инструменты для построения конфигурируемых моделей шума для реалистичной симуляции ошибок, возникающих во время вычислений на реальном квантовом оборудовании.
- Qiskit Ignis – это компонент, содержащий инструменты для измерения и верификации уровня шума в устройствах ближнего действия, а также позволяющий проводить вычисления в присутствии шума. Также следует отметить наличие процедур смягчения шума, которые реализованы в виде калибровочных схем и могут быть применены к наборам результатов, полученных с использованием одной и той же платформы.

Главным преимуществом Qiskit является возможность бесплатно проводить вычисления на суперкомпьютере IBM с предоставлением 5 кубитов. Для этого требуется создать аккаунт IBM Q Experience и получить токен для дальнейшего доступа. Рассмотрим пример на языке программирования Python:

```
from qiskit import QuantumCircuit, QuantumRegister, ClassicalRegister, execute
from qiskit import IBMQ
from qiskit.providers.ibmq import least_busy
IBMQ.enable_account('PASTE_YOUR_API-KEY_HERE')
least_busy_device = least_busy(IBMQ.get_provider().backends(simulator=False))
q = QuantumRegister(1)
c = ClassicalRegister(1)
state = QuantumCircuit(q, c)
state.measure(q, c)
state.draw(output='mpl')
job = execute(state, backend=least_busy_device, shots=1024)
import time
while job.status().name != 'DONE':
   print(job.status())
   time.sleep(10)
result = job.result()
from qiskit.tools.visualization import plot_histogram
plot_histogram(result.get_counts(state))
```

После импорта всех необходимых сущностей активируется аккаунт с использованием полученного ранее токена, находится наиболее доступное устройство для проведения вычислений, создаются квантовый и классический регистры и соответствующая квантовая схема, на которую добавляется гейт измерения.

Команда *state.draw(output='mpl')* позволяет отобразить текущее состояние схемы. В данном случае она будет выглядеть так, как показано на рис. 1.

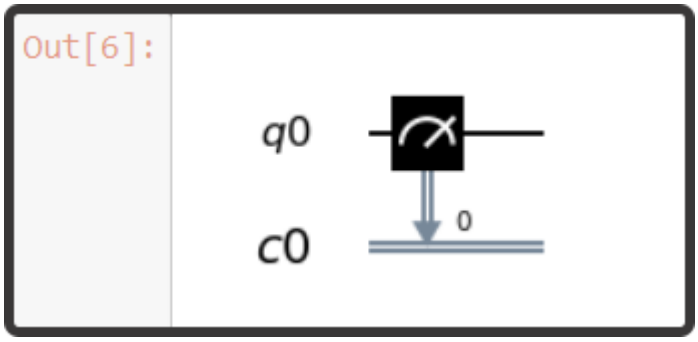

Рис. 1. Отображения состояния квантовой схемы
Fig. 1. Quantum scheme image

Далее создается задание на выполнение схемы (функция *execute*) на реальном устройстве, и с периодичностью 10 секунд проверяется статус выполнения задания. Когда задание создано, оно помещается в очередь со статусом QUEUED, с началом выполнения статус сменяется на RUNNING, а по завершении принимает значение DONE.

Визуализировать результат можно с использованием столбчатой диаграммы. На рис. 2 показан примерный вывод состояния кубита.

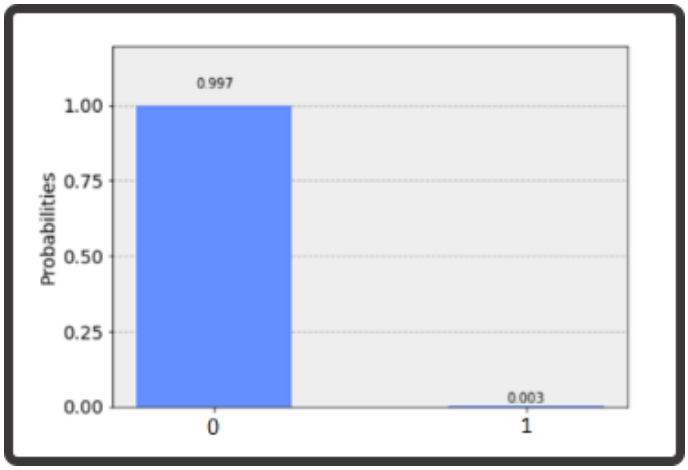

Рис. 2. Визуализация результатов
Fig. 2. Results Visualization

Таким образом, главной особенностью Qiskit, безусловно, является уникальная возможность проведения вычислений на действительно существующем квантовом устройстве, а не только с использованием локального эмулятора. Однако у данного преимущества есть обратная сторона – ждать начала выполнения задачи приходится довольно долго, в данном случае время ожидания составило порядка 9 минут.

Также достоинствами являются открытый исходный код и подробная документированность – на официальном сайте можно найти инструкции по установке фреймворка и примеры использования.

4. **Quantum Programming Studio.** Данный сервис представляет собой веб-интерфейс, который позволяет создавать квантовые схемы и алгоритмы, а после получать результаты путем непосредственного моделирования в окне браузера. При этом вычисления могут осуще-

ствляться, как в режиме эмуляции с помощью встроенного пакета Quantum cirquit (был рассмотрен ранее), так и непосредственно на квантовом компьютере [8].

Quantum Programming Studio (QPS) позволяет конструировать квантовые схемы путем простого перетаскивания ее элементов из специальной панели инструментов, что показано на рис. 3.

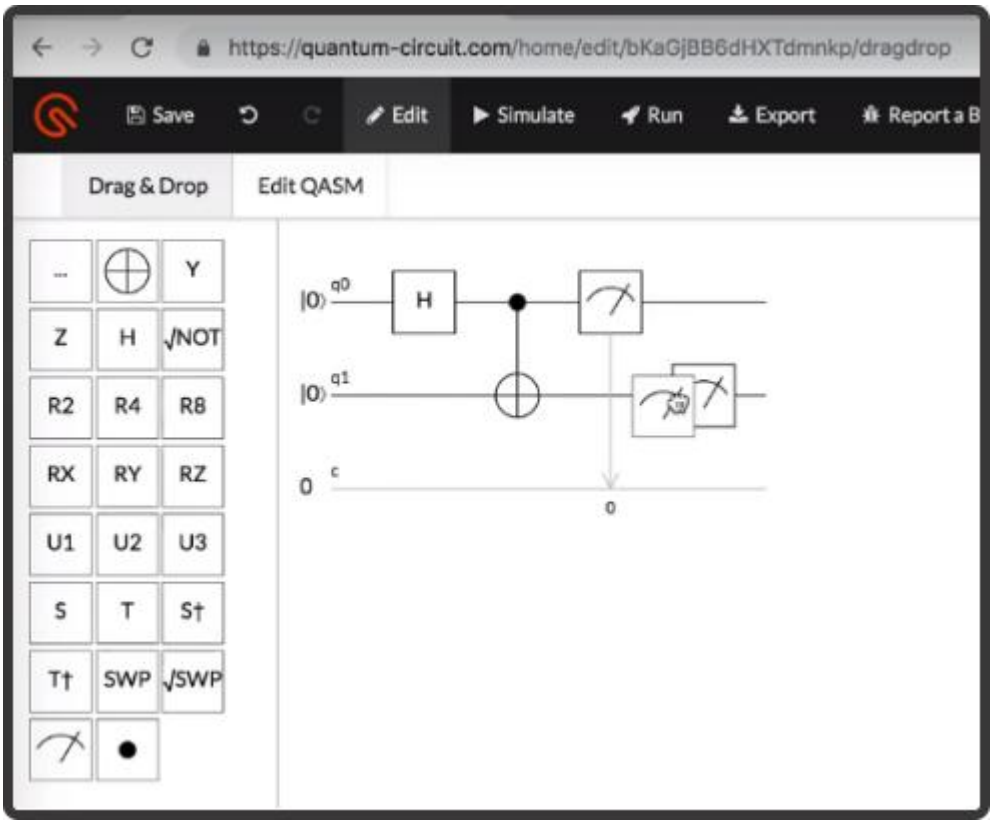

Рис. 3. Создание квантовый схемы в QPS
Fig. 3. Creating quantum circuits in QPS

Каждый клиент сервиса QPS взаимодействует лишь с его графическим веб-интерфейсом. Непосредственные вычисления могут происходить на сторонних сервисах и эмуляторах и в редких случаях на клиентской ЭВМ. QPS позволяет работать с Rigetti QCS, Rigetti Forest SDK и IBM Qiskit на выбор. Эти сервисы позволяют выполнять вычисления на своих квантовых компьютерах и специальных эмуляторах [7]. Обычно пользователь QPS взаимодействует с веб-интерфейсом локально у себя на ЭВМ посредством HTTPS протокола. Также и выполнение квантовых схем происходит либо на пользовательском компьютере, либо в облаке, где установлены все необходимые программные зависимости. Для этого в процессе работы происходит установка соединения через защищенный веб-сокет между сервером QPS и QPS-клиентом, который выполняет квантовые схемы на серверном симуляторе или же на оборудовании партнерских компаний. QPS-клиент является адаптером между запросами пользователя и непосредственно реальным оборудованием, которое предоставляют партнеры QPS. Схема полного взаимодействия пользователя с сервисом представлена на рис. 4.

[8] Quantum Programming studio, 2020. URL: https://quantum-circuit.com/docs (дата обращения 27.03.2020).

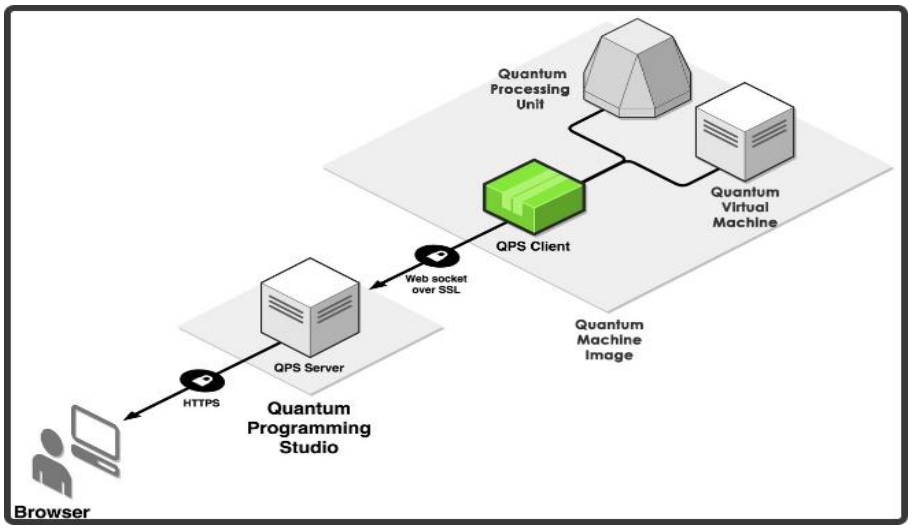

Рис. 4. Взаимодействие пользователя с сервисом QPS
Fig. 4. User Interaction with QPS

Отмечая все особенности QPS, можно выделить ряд достоинств и недостатков. К достоинствам, несомненно, можно отнести возможность выполнения вычислений в облаке в режиме эмулятора и, конечно же, тот факт, что QPS может работать с разными реализациями настоящих квантовых компьютеров. Здесь также стоит сказать о кросс-платформенности и гибкости всего сервиса QPS: пользователь сможет разрабатывать и выполнять свои квантовые решения с абсолютно любого устройства и из любой точки мира. Недостатком по сути является тот факт, что если пользователь захочет использовать локальные эмуляторы, то ему придется устанавливать их самостоятельно.

5. **Microsoft Quantum Development Kit (MQDK).** Выше были рассмотрены различные конкретные решения для работы с квантовыми алгоритмами и компьютерами, а MQDK представляет собой целую инфраструктуру, которая может пригодиться как начинающему, так и уже опытному разработчику в области квантовых технологий.

Основным инструментом для всех инструментов MQDK служит квантово-ориентированный язык программирования Q# и библиотеки для него, которые находятся в полностью открытом доступе [9]. С их помощью даже новичок сможет построить свое квантовое решение. Доступен Q# как составляющий элемент загружаемого для Visual Studio плагина MQDK. Данный язык также считается новаторским в области квантовых компьютеров, предоставляет весь инструментарий разработчика, включая полноценную отладку написанных приложений и возможность оценить затраты на запуск решения. На данный момент Q# способен в условиях эмуляции работать с 30 кубитами. Стоит отметить, что Q# совместим с языком Python. Эта совместимость по сути заключается в наличии модуля qsharp для Python, который позволяет запускать программы, написанные на языке Q# [10].

Конечно же, MQDK должен обладать возможностью использовать для вычисления квантовых решений настоящие квантовые компьютеры и специализированное высокопроизводительное программное обеспечение. С этой целью создан сервис Azure Quantum, который представляет собой огромный набор служб, включает готовые квантовые решения, программное обеспечение, способное достичь связности порядка 40 кубитов. Azure Quantum

[9] Microsoft Quantum Documentation / Microsoft Docs, 2020. URL: https://docs.microsoft.com/en-us/quantum/ (дата обращения 08.04.2020).
[10] Там же.

представляет своим пользователям доступ к наиболее конкурентно способным предложениям на рынке квантовых технологий. Данный сервис дает возможность использовать инструменты, которые можно разделить на следующие категории:

- готовые квантовые решения, работающие в промышленных масштабах;
- квантовое ПО (симуляторы, средства оценки ресурсов);
- квантовая аппаратная система с множеством конфигураций связанных кубитов;
- инструменты масштабирования, безопасности и постоянной поддержки от команды Azure и их партнеров.

Структура всей системы Azure Quantum представлена на рис. 5.

Разработчики MQDK также позаботились о теоретической и практической подготовленности своих пользователей. Благодаря коллекции учебных пособий под названием «Quantum Katas» каждый может изучить на наглядных примерах принципы квантового программирования и научиться работать со всей инфраструктурой MQDK. Также на официальном сайте MQDK можно найти ссылку на GitHub репозиторий с всевозможными рабочими примерами квантовых решений [11].

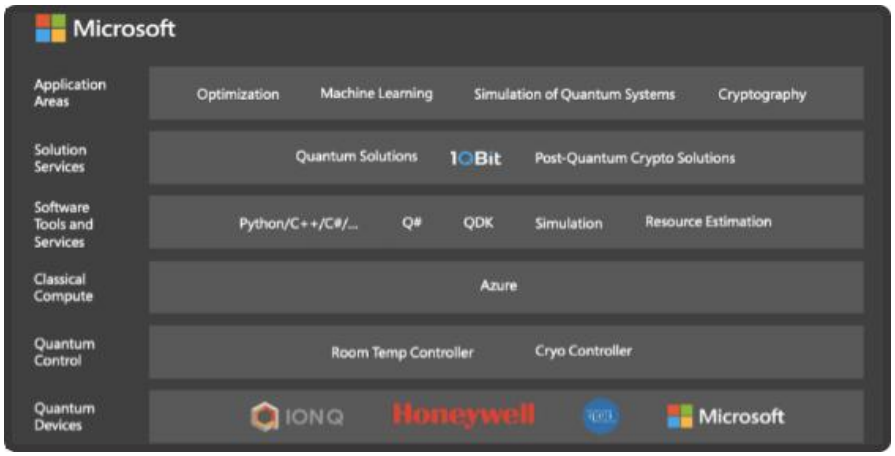

Рис. 5. Стек сервиса Azure Quantum
Fig. 5. User Interaction with QPS

В целом MQDK является наиболее оптимальным решением для разработки и изучения квантовых решений. Уже на данный момент MQDK – это набор очень удобных инструментов, которые помогут овладеть знаниями в области квантовых вычислений, научиться разрабатывать и оперировать квантовыми решениями.

Заключение. В данной работе рассмотрены различные библиотеки для проведения квантовых вычислений, проведено их сравнение и проанализированы достоинства и недостатки. Каждый из инструментов предназначен для использования с различным языком программирования, однако все они имеют открытый исходный код и распространяются бесплатно.

Возможности библиотек в целом схожи, однако Quest и Qiskit выделяются на общем фоне: первая благодаря поддержке распределенных вычислений, вторая из-за возможности использования 5 кубит квантового суперкомпьютера IBM для расчета схем. Конечно, на данный момент реализация технологии имеет недостатки, но сама возможность подключиться к квантовому компьютеру, пусть и в учебных целях, уникальна.

Выбор конкретного инструмента каждый пользователь должен осуществлять сам, в зависимости от своих целей и предпочтений. Авторам данной статьи наиболее импонирует биб-

[11] Microsoft Quantum Documentation / Microsoft Docs, 2020.

лиотека Qiskit благодаря возможности создавать квантовые схемы на высоком уровне абстракции, хорошей документированности и использования имеющего низкий порог вхождения языка Python в качестве основной платформы.

Список литературы

1. **Манин Ю.** Вычислимое и невычислимое. М.: Сов. радио, 1980. 128 с.
2. **Feynman R. P.** Simulating physics with computers. *International Journal of Theoretical Physics*, 1982, no. 21, p. 467–488. DOI 10.1007/BF02650179
3. National Academies of Sciences, Engineering, and Medicine. Quantum Computing: Progress and Prospects. Washington, DC, The National Academies Press, 2018. DOI 10.17226/25196
4. **Aryte F., Arya K., Babbush R. et al.** Quantum supremacy using a programmable superconducting processor. *Nature*, 2019, no. 574, p. 505–510. DOI 10.1038/s41586-019-1666-5
5. **Jones T., Brown A., Bush I., Benjamin S. C.** QuEST and High Performance Simulation of Quantum Computers. *Scientific Reports volume*, 2019, no. 9. DOI 10.1038/s41598-019-47174-9
6. **Moran C.** Mastering Quantum Computing with IBM QX: Explore the world of quantum computing using the Quantum Composer and Qiskit. Packt Publishing, 2019. 308 c.
7. **Kaiser S. C., Granade Ch. E.** Learn Quantum Computing with Python and Q#. Manning Publication, 2020. URL: https://www.manning.com/books/learn-quantum-computing-with-python-and-q-sharp#reviews (дата обращения 03.04.2020).

References

1. **Manin Yu.** Computable and non-computable. Moscow, Soviet Radio Publ., 1980, 128 p. (in Russ.)
2. **Feynman R. P.** Simulating physics with computers. *International Journal of Theoretical Physics*, 1982, no. 21, p. 467–488. DOI 10.1007/BF02650179
3. National Academies of Sciences, Engineering, and Medicine. Quantum Computing: Progress and Prospects. Washington, DC, The National Academies Press, 2018. DOI 10.17226/25196
4. **Aryte F., Arya K., Babbush R. et al.** Quantum supremacy using a programmable superconducting processor. *Nature*, 2019, no. 574, p. 505–510. DOI 10.1038/s41586-019-1666-5
5. **Jones T., Brown A., Bush I., Benjamin S. C.** QuEST and High Performance Simulation of Quantum Computers. *Scientific Reports volume*, 2019, no. 9. DOI 10.1038/s41598-019-47174-9
6. **Moran C.** Mastering Quantum Computing with IBM QX: Explore the world of quantum computing using the Quantum Composer and Qiskit. Packt Publishing, 2019. 308 c.
7. **Kaiser S. C., Granade Ch. E.** Learn Quantum Computing with Python and Q#. Manning Publication, 2020. URL: https://www.manning.com/books/learn-quantum-computing-with-python-and-q-sharp#reviews (дата обращения 03.04.2020).

Материал поступил в редколлегию
Received
03.05.2020

Сведения об авторах

Баскаков Павел Евгеньевич, студент 5-го курса кафедры «Кибербезопасность информационных систем» Донского государственного технического университета (Ростов-на-Дону, Россия)
pavelbaskakov98@gmail.com

Хабовец Юрий Юрьевич, студент 5-го курса кафедры «Кибербезопасность информационных систем» Донского государственного технического университета (Ростов-на-Дону, Россия)
yuriy131196@gmail.com

Пилипенко Ирина Александровна, ассистент кафедры «Кибербезопасность информационных систем» Донского государственного технического университета (Ростов-на-Дону, Россия)
irenphil@yandex.ru

Кравченко Вероника Олеговна, аспирант 3-го года кафедры «Кибербезопасность информационных систем» Донского государственного технического университета (Ростов-на-Дону, Россия)
olenikr@yandex.ru

Черкесова Лариса Владимировна, доктор технических наук, профессор кафедры «Кибербезопасность информационных систем» Донского государственного технического университета (Ростов-на-Дону, Россия)
chia2002@inbox.ru

Information about the Authors

Pavel E. Baskakov, 5th year student of the Department “Cybersecurity of information systems” of the Don State Technical University (Rostov on Don, Russian Federation)
pavelbaskakov98@gmail.com

Yuri Yu. Khabovets, 5th year student of the Department “Cybersecurity of information systems” of the Don State Technical University (Rostov on Don, Russian Federation)
yuriy131196@gmail.com

Irina A. Pilipenko, assistant of the Department “Cybersecurity of information systems” of the Don State Technical University (Rostov on Don, Russian Federation)
irenphil@yandex.ru

Veronika O. Kravchenko, 3-year post-graduate student of the Department “Cybersecurity of information systems” of the Don State Technical University (Rostov on Don, Russian Federation)
olenikr@yandex.ru

Larisa V. Cherkesova, doctor of technical Sciences, Professor of the Department “Cybersecurity of information systems” of the Don State Technical University (Rostov on Don, Russian Federation)
chia2002@inbox.ru