SN Computer Science (2024) 5:410
https://doi.org/10.1007/s42979-024-02761-0

ORIGINAL RESEARCH

Performing Distributed Quantum Calculations in a Multi-cloud Architecture Secured by the Quantum Key Distribution Protocol

Jose Luis Lo Huang[1] · **Vincent C. Emeakaroha**[1]

Received: 28 November 2023 / Accepted: 29 February 2024

Abstract

Quantum computing (QC) is an emerging area that yearly improves and develops more advances in the number of qubits and the available infrastructure for public users. Nowadays, the main cloud service providers (CSP) are implementing different mechanisms to support access to their quantum computers, which can be used to perform small experiments, test hybrid algorithms and prove quantum theories. Recent research work have discussed the low capacity of using quantum computers in a single CSP to perform quantum computation that are needed to solve different experiments for real world problems. Thus, there are needs for computing powers in the form of qubits from multi-cloud environment. Quantum computing in a multi-cloud environment requires security of the communicating channels. A well known algorithm in quantum cryptography for this purpose is the quantum key distribution (QKD) protocol. This enables the sender and receiver of a message to know when a third party eavesdropped any data from the insecure quantum channel. To address the low capacity issue, this research develops and tests the use of heterogeneous quantum computers located on different CSP to distribute quantum calculations between them by leveraging the channel security provided by the QKD protocol. The achieved results show over 88.1% of correct distributed quantum computation results without error correction methods, 96.8% of correct distributed quantum computation results using error correction methods and over 98.8% correct authorisation detection in multi-cloud environments. This demonstrates that quantum calculations can be distributed between different CSP while securing the channel with the QKD protocol at the same time.

Keywords Multi-cloud · Quantum computing · QKD protocol · Data lake · Quaternions · Cloud computing

Introduction

In recent times, cloud computing has become a matured area where companies and users can run their workloads leveraging on flexible cloud service provider (CSP) infrastructures and paying only for what they actually use in an on-demand consumption and payment method. Currently the main CSP such as Amazon Web Services (AWS)[1], Microsoft Azure[2], Google Cloud (GCP)[3] or IBM Cloud[4] are offering multiple services to the public that enable an exponential advance in the technology industry, as the users now can focus more on the business use cases and less on the infrastructure set up. Among the services offered by these CSP is the quantum computing scheduling and runtime. Research has shown the inability of a single CSP to provide enough quantum computing power for meaningful real world experiments [1]. Therefore, efforts have been geared towards the use of multiple cloud infrastructures to achieve optimised quantum computation.

Moreover, quantum computing (QC) is having significant advances based on prototypes of quantum machines on cloud infrastructures that can execute workloads that would take a large amount of time in conventional computers [2,

This article is part of the topical collection "Recent Trends on Cloud Computing and Services Science" guest edited by Claus Pahl and Maarten van Steen.

✉ Vincent C. Emeakaroha
vincent.emeakaroha@mtu.ie

Jose Luis Lo Huang
07-41108@usb.ve

[1] Department of Computer Science, Munster Technological University, Cork, Munster, Ireland

[1] https://aws.amazon.com/.
[2] https://azure.microsoft.com/en-us/.
[3] https://cloud.google.com/.
[4] https://www.ibm.com/cloud.

Published online: 08 April 2024

3]. A quantum bit (or qubit) is the smallest unit of quantum information, which is usually represented by an atom's state, electron, photon or other elementary particle. Unlike a classical bit, a quantum bit can exist in superposition states (both 0 and 1 at the same time with different probabilities) and have more features that permit different approaches not possible with classical computation [4].

With the initial release of these quantum computers located on the different CSP, it is now possible to execute and to test a good number of protocols and algorithms that before only existed in theory. The quantum key distribution (QKD) protocol is a specialized quantum algorithm that permit to securely send data between two entities through an insecure link. With this protocol, they can detect if someone is trying to capture or is measuring the information that traverse the insecure channel.

A good number of researches use quantum computers from one CSP to perform calculations based on hybrid algorithms, for example [5, 6]. However, there is a strict limitation in the number of qubits on each quantum computer. This restrict the possibility to execute bigger experiments that needs more quantum power or qubits. Also, current research efforts are focusing on integrating the QKD protocol into cloud deployments to secure the communications between a single cloud provider and users. However, recent trends have shown the use of multi-cloud deployments to provision services that access data from distributed sources. This trend has brought security issues around distributed computations. The key issues are (1) low number of qubits in a single CSP quantum environment is not helpful in running bigger quantum workloads, (2) insecurity in the distributed calculations due to poor parameter configurations, and (3) interruption by third party if incorrect methods are used in the communication channel.

This research proposes a mechanism to distribute quantum calculations between a multi-cloud architecture, consisting of different CSPs while improving the security of their communication channels using the quantum key distribution (QKD) protocol. The achieved results show over 88.1% of correct distributed quantum computation results without error correction methods, 96.8% of correct distributed quantum computation results using error correction methods and over 98.8% correct authorisation detection in multi-cloud environments.

The rest of the paper is organised as follows: Sect. 2 discusses the background and related work. In Sect. 3, we discuss the design of the work. Implementation details are provided in Sect. 4. Next, the evaluations are presented in Sect. 5. Finally, Sect. 6 concludes the paper.

Background/Research Context

This section highlights key background information and discusses related work.

Background

Currently, more than 90% of companies have adopted cloud computing as shown in recent surveys [7, 8]. Moreover, companies are increasing the cloud first strategies and cloud native services or technologies. Also, as stated in the same references, the multi-cloud approach is gaining more adepts each year, with more than 10% by 2021, and more than 35% expecting to use it in 2024. This is usually to avoid a CSP or vendor lock-in, and to add backups or redundancy to their services or components between different and distributed sites. Others are using it to reduce the costs by splitting the applications and infrastructures depending on the price and performance average, for example using Windows machines on Azure, but Aurora databases from AWS.

Modern quantum computing researches began with Paul Benioff in [9] where he constructed a microscopic quantum mechanical model of computers as represented by Turing machines. Later, Richard Feynman suggested in his work [10] that a hypothetical universal quantum computer should be able to simulate quantum physics. Then, many advances have been done on this area, including important algorithms, such as Shor's algorithm [11], which with the correct technology should be able to break RSA encryption and the whole internet secure communications and transactions. He found a way to speed up the calculation of large prime factors of an integer, which is the base of the well known RSA and SSL security protocols. Then, when a powerful enough quantum computer will be released that will be capable of execute this kind of calculation, the security mechanisms used today won't be secure anymore.

Today there are a good number of quantum computers hosted on the main cloud service providers and other private companies. They are still in early stages and the experts [12] forecast a 8.6 billion dollar market growth in 2027 on this research area. The main CSPs that offer quantum computing services are IBM, AWS, GCP and Azure. Currently, IBM is the one with the biggest quantum machine in terms of qubits quantity with 433 qubits on an Osprey chip.[5]

In classical computation, a bit is the smallest form of information representation. In any specific time, a bit can only be 0 or 1. Meanwhile, in quantum computation, a qubit will result in 0 or 1 as well when measured, however before the measurement process it could be 0 and 1 at the same time

[5] https://www.ibm.com/quantum/systems.

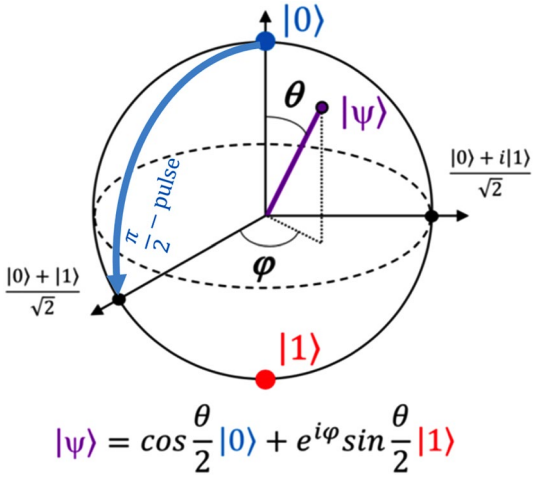

Fig. 1 The Bloch sphere

with different probabilities. This is a direct consequence of the applied quantum mechanics participation.

To mathematically represent this behaviour, the most used representation is the Dirac notation [13]. With this, the 0 qubit and the 1 qubit can be represented as vectors as follows:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{1}$$

$$|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{2}$$

In the zero state (1), when the result is $|0\rangle$, it could be interpreted as all the probabilities are in the position 0 of the vector. And in the one state (2), when the result is $|1\rangle$, all the probabilities are in the position 1 of the vector.

The graphic representation of a bit could be a switch set to on or off meaning 0 or 1. Meanwhile, to represent a qubit the usual way is by using a bloch sphere [14] (Fig. 1, [15]) where the highest point is the $|0\rangle$, the lowest point is the $|1\rangle$ and every point between them is a combination of probabilities between these two values.

One of the most known quantum protocols in quantum cryptography is the QKD protocol defined in 1984 by Bennett and Brassard in [16]. This protocol consists of using a quantum channel to send a one pad secret message that will be deformed and being noticed by both sender and receiver if it is eavesdropped by a third party due to the physical properties of photons. Using the QKD that consists of sending a number of random qubits, and a number of operations, the receiver will measure those and will share parts of the results. If someone has measured the data before, then the results from the measures in the target endpoint will be incorrect. We use this protocol in this work to implement secure communication channel for the multi-cloud usage senarios.

Related Work

The quantum operations based on quantum cloud computing services have been a trending topic in the last years. In [17], the coupling between the two worlds of cloud computing and quantum computing is shown to understand how they are becoming a strong team by integrating both technologies. Moreover, in [18], they provided an extended work studying the effects of quantum jobs (mostly in IBM quantum machines) and compared several machine versions with other high performance computers (HPC) based on compilation times, fidelity, queuing times and execution times to analyze the trends and the state of quantum machines in the cloud. Also, in [19], the researchers provided a mechanism to reduce energy consumption by 23.36% and makespan 20% on average by using a multi-objective quantum-inspired genetic algorithm (MQGA) when using quantum computers with high computation workloads. Another example is offered in [20], where they proposed a new priority queue mechanism for the quantum cloud computing offerings to achieve a better quality of service (QoS) based on the urgency of the workload. Based on these articles, it is proven that there is an important audience in the middle of both cloud computing and quantum computing that is running their workloads in the current quantum cloud computing existing services.

Major improvements has been proposed in quantum cloud computing operations in recent researches. For example, in [21] the researchers proposes a new approach for mapping quantum programs in the cloud environment to avoid waste of resources by using a X-SWAP scheme that enables inter-program SWAPs and prioritizes SWAPs associated with critical gates to reduce the SWAP overheads improving the compilation times by 11%. Similarly, in [22], a qubit routing procedure that uses a modified version of the deep Q-learning paradigm is proposed to minimise the circuit depth added by the SWAP gates. Also, in [23], the authors exploited circuit slack for single-qubit gates that occur in idle windows, scheduling the gates such that their timing can counteract some errors. Thus, improving the error correction and the fidelity of the results. Moreover, in [24] an improvement is proposed to reduce the number of gates of a random quantum circuit by $\sim$ 30% by using an algorithm for pattern matching that finds all maximal matches. As can be seen, all these propositions were made to improve a quantum environment running on a single CSP. However, there are not proposals to improve the latency and calculations times by splitting the workload in a multi-cloud alternative.

Only few researches were found that can be related with the use of multi-cloud quantum systems. One is the work in

[25], which targets to combine the portable XACC quantum programming frontend and the scalable ExaTN numerical processing backend. They introduced an end-to-end virtual quantum development environment that can scale from laptops to future exascale platforms. This is not directly related to multi-cloud infrastructures, but can be used to deploy in the future with some modifications. In the other hand, in [26], the researchers used blockchain networks (BCN), based on multi-cloud architectures, to track the operational steps of network service instantiation metrics while benefiting from the security features of post-quantum cryptography (PQC). Using a N-th degree truncated polynomial ring units (NTRU), they had shown that Quorum can provide a lower average time-to-write value compared to other BCNs considered (Ethereum and Hyperledger). Therefore, is evident that the works in quantum security for quantum multi-cloud architectures to distribute quantum calculations is still in its initial stages and much research is needed to reach a mature state.

In terms of cloud data transmission, in [27], the authors created a hybrid quantum protocol using 3DES and QKD to demonstrate the viability of secure dual party communication between two resources in the cloud. They did experiments using a simulator for the BB84 part that must run in a quantum channel. Similarly, in [28], the authors do a performance analysis of QKD network structures suitable for power business scenarios. They proposed and did simulations to test a quantum VPN to enhance the network security. They confirmed that for shorter quantum signal state correction time, the quantum key rate efficiency was higher. In [29], the researchers uses a modified version of the BB84 combined with the elliptic curve digital signature to secure cloud data transmission. However, comparing these works to this research, they provided no methods to improve the data access, authorisation and transmission security on a multi-cloud environment taking advantage of the current quantum technologies, specifically the QKD protocol.

To the best of our knowledge, none of the aforementioned research work investigated the execution of distributed quantum calculations based on a multi-cloud environment including error correction methods considering data transmission secured by the QKD protocol.

Design

This section discusses the design of our proposed solution. Before going deep into it, we first highlight the key challenges to be addressed in this paper for distributed quantum calculations, which are:

- The quantum computers has an insufficient number of qubits to run big experiments.

  For big quantum workloads, usually the method to be used is to split and run them sequentially due to the small number of qubits in the currently existing quantum computers.
- The multi-cloud data access and authorization process is insecure.

  The data transmission between the CSP pass through the Internet and other unreliable networks, and can be eavesdropped.

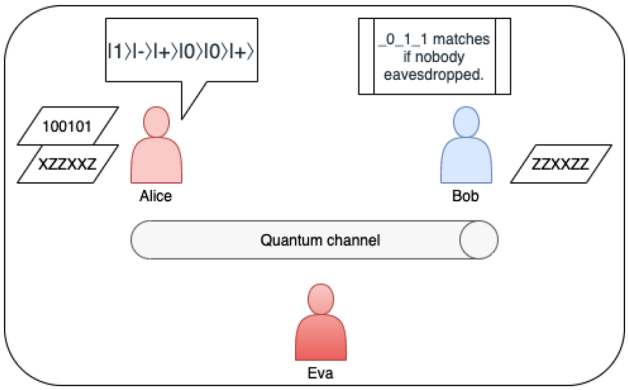

Fig. 2 QKD protocol example

The QKD protocol is well known as a one-pad security process, as discussed previously. Figure 2 shows an example of the components of the protocol and the working steps.

The steps are as follows:

- Step 1: Alice chooses a string (the message) and a random choice of basis or operations. Those are private to Alice. For example:

  $string = 100101;$
  $basis = XZZXXZ;$

  where X means a bit and a Z means an operation.
- Step 2: Alice encodes each bit onto a string of qubits using the chosen basis and send it to Bob. This means that each qubit is in one of the following states $|0\rangle$, $|1\rangle$, $|+\rangle$ or $|-\rangle$ chosen at random. For example:

  $message = |1\rangle|-\rangle|+\rangle|0\rangle|0\rangle|+\rangle$
- Step 3: Bob measures each qubit randomly using a chosen basis. For example:

  $basis = ZZXXZZ$
- Step 4: Alice and Bob publicly share the basis they used for each qubit. If Bob measured a qubit on the same basis that Alice has prepared it, then they use this to form part of the shared secret key, if not, this qubit is discarded. For

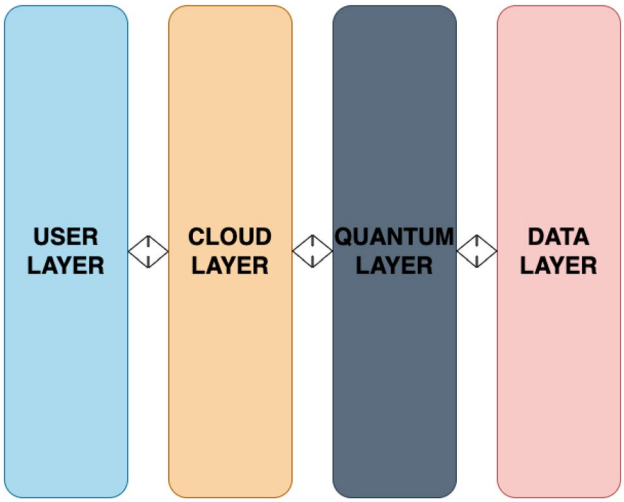

Fig. 3 The main layers of the project

this example, the basis are equal in the second, fourth and sixth position (_Z_X_Z), then the value on those positions should be the same for both (_0_1_1) and will be part of the key to share.

- Step 5: Finally, both share a random sample of their keys (which after the previous process should be 011), and if the samples match, they can be in a highly probability sure that their transmission was not eavesdropped by a third party (Eva in the figure).

By extracting the first bytes of each file, instead of using all the file, to validate the transmission, we improve the speed and the overall processing of data distribution could be faster. The security will prevail with the QKD protocol and even, would be enhanced as the algorithm has been researched and tested previously in several situations [30]. It will detect any eavesdropping on the network if it happens.

In this research, we aim to use the QKD protocol to enhance the data authorization and the transmission security of data that are distributed in multiple clouds (multi-cloud). To accomplish this, the use of the different quantum computers from each CSP could help by running the quantum sections of the QKD protocol.

To address error corrections, we tested simple mechanisms, such as repetition codes [31]. They were tested with both quantum hardware and quantum simulators. The results on quantum simulators were better reaching 8% more fidelity. The use of Shor codes [32] was also taken into consideration in the experiments, but the results were different according to the underlying quantum hardware used. Therefore it needs further investigation and tests to provide better results.

In designing our solution, we divide it into four main layers: the user layer, the cloud layer, the quantum layer and the data layer. In Fig. 3, we show how the main layers should communicate between themselves and therefore how the data will flow among them. In the next lines, we discuss the responsibilities of each layer.

- User layer
  This layer is composed by the user of the system and the programs and hardware that are executed locally on the user computer.
- Cloud layer
  All the resources that belongs to any of the CSPs belongs to this layer. For example, the identity, compute and networking resources that will be used.
- Quantum layer
  This layer includes all the quantum computers (QC), simulators and any software that is pure quantum or hybrid.
- Data layer
  Although the resources on this layer also are on the cloud, the data layer can be taken as a different section, as it is the final destination of the data files.

The distributed data files to be used are expected to be composed of uncommon elements that will be split between the CSPs. The quaternions are an extension of the complex numbers and were first defined by the Irish mathematician sir William Hamilton [33]. The data to be used for the distributed quantum computations will be Hamilton's quaternions. Many small calculations and huge matrices are based on those.

The quaternions can be represented as a linear combination as shown in the formula (3)

$$H = \alpha + \beta i + \gamma j + \delta k \tag{3}$$

To represent quaternions in software, the inputs needed are the values for α, β, γ and δ. This data should be able to be generated on the user layer or on the cloud layer.

In Fig. 4, we show the general multi-cloud architecture to be used to determine the data transmission between the different CSPs when splitting the distributed data. Similar approaches will be used for data access and authorization. But instead of the data in the data lakes, the messages to be delivered as quantum blocks will be the user data.

In general, the proposed solution is designed to use the following steps, which will be discussed in details later on:

- Step 1:
  The quaternion data will be generated from a program developed during the research, then this data will be stored on files with size up to 2GB.
- Step 2:

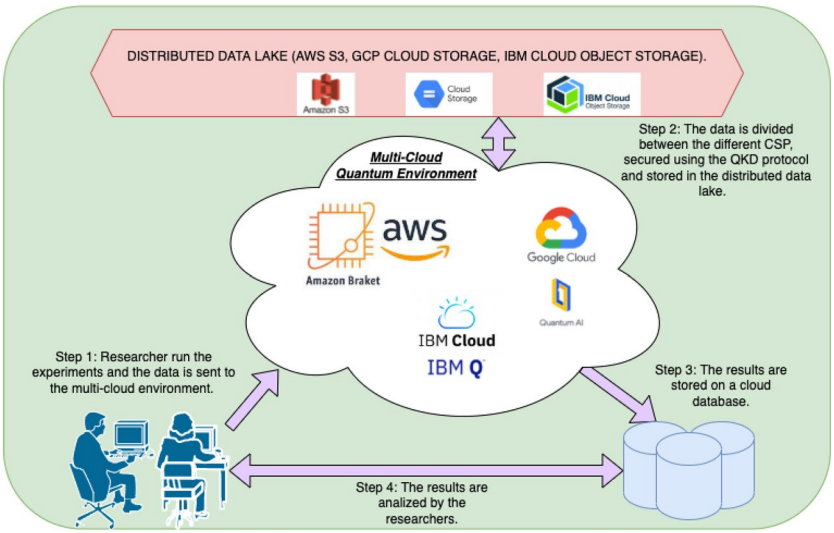

Fig. 4 Multi-cloud distributed quantum data calculation and data lake architecture

Those files will be split and distributed onto the different CSP by a second program. Once there, the data will be processed by the quantum computers to execute the QKD tests and the validations before being stored on the native data lake.

- Step 3:
  In parallel, another program will take the times and evaluate the outputs from each CSP and check the times. In summary, the following are the main evaluation outputs from each test:
  - The duration of each file upload, distribution and storage process.
  - The QKD validation output.
  - The hash of each file fragment.

  It is also expected to evaluate the complexity of the algorithms developed during the investigation. One objective is to tune the algorithms to achieve a upper bound of logarithmic or linear time. The latter may or may not be possible.
- Step 4:
  The data files stored on the data lakes are used as input for the quantum computers in each CSP and the calculations are made, which output are then sent to both the data lake and a central hub, which will merge and process the results accordingly to confirm accuracy and correctness.
- Step 5:
  In this final step, we will review the achieved results and plot some graphics to confirm and demonstrate the preposition.

Next, we provide some details to the design steps.

Data Generation

The first step is the data generation. This step is expected to be able to run both locally and in a cloud VM. This cloud VM should be a hub between the users and the multi-cloud quantum data lake system. It is expected to have an autoscaling group (ASG) for this cloud hub VM components. This is to increase the compute power if needed by the program when generating the calculation files. Moreover, the architecture includes two availability zones (AZ) for high availability purposes.

Data Distribution

Once the data was generated (if it was generated on the local machine, it must be transmitted to the cloud hub VM before this step), the next step is to distribute the data files. For this the cloud hub ASG must have proper connections to all the CSPs. The cloud hub ASG must have proper connections to the 3 CSPs that will be used for the implementation. Additionally, a cloud database (DB) must be running in the same VPC as the cloud hub ASG. On this cloud DB, all the results will be written and also it will be queried by the researcher to get information related with times and correctness. Therefore, the cloud DB should have one instance in each AZ.

Multi-cloud Quantum System

Once the data files enters on this stage, they will be processed by each CSP in parallel. Each one running its native quantum code with the different quantum API (Braket for

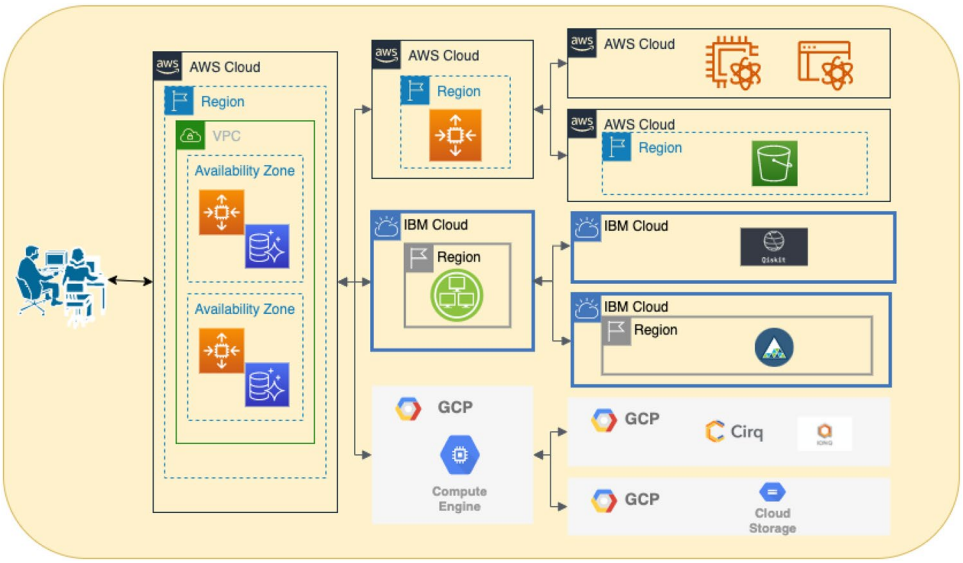

Fig. 5 Full system design

AWS, Qiskit for IBM and Cirq for GCP). It will use a hybrid algorithm to produce both the distributed quantum calculations and the QKD implementation.

Full System Design

The full system includes all the previously mentioned components. The data files are generated locally or in the cloud VM hub, then they are distributed to each CSP server using the internal network from each CSP. Ideally it should use a quantum network as the channel. Once the data reach this stage, the header of each data file is verified using the QKD protocol and compare it to the one in the hub, if the verification is successful, then the data file is stored on the local data lake service. Once they are in their respective data lake service, they are used as inputs for the quantum calculations in each CSP executing the hybrid code including error correction methods, which then returns the results to both the data lake and the hub.

Figure 5 presents the complete system design. All the previous components are added on this figure. It can be clearly seen how the data flows throughout the system. It will go from the user to the cloud hub ASG, and then to the server on each CSP, and from this server the calculations are made using the QKD protocol on the quantum machines and simulators before authorizing the data files to be stored on the local data lake on each CSP.

Implementation

This section describes the implementation details of our proposed solution. We called the first full version of this work QLake, as it is the first known Quantum technology based Data Lake which distribute quantum calculations between heterogeneous quantum computers located in different CSP and store the solutions in a distributed data lake.

For this work, the use of GitHub was important to maintain the code on the cloud, and to get all the benefits from this platform, such as change management, code history, bug reports, etc. Also, it will provide the open source code to the different quantum and cloud developer communities, who can take advantage of it and extend the purpose and use cases. Part of the code of this research is on GitHub [34].

Cloud Hub VM

The VM used in AWS as the hub node is an Ubuntu Server 20.04 LTS (HVM) with SSD Volume Type and AMI ID ami-04505e74c0741db8d (64-bit x86). The version 20.04 was the latest LTS at the moment of the experiments, and this was the reason for the choice. The instance type is m5.large (2 vCPUs and 8 GB memory), which is a general use instance type with standard size. The packages that are installed on this VM for the data generation phase are: Python 3.10.2, Anaconda 3, NumPy, SciPy, Numba, quaternion.

Multi-cloud Environment

To implement the solution for this research, the CSPs with the most mature quantum services at that moment were used. These CSPs were AWS, IBM Cloud and GCP.

In the implementation of the quantum layer, it is important to note that a global quantum network does not exists at the moment due to the lack of advanced mechanisms. Although there are regional advances in this subject [35], a global approach is in experimental analysis by many researchers and hopefully a global quantum network will

exists very soon. For this reason, the quantum communication between the cloud hub VM and the quantum machines was done as expected (directly), but the communication between the different quantum machines was made by using the cloud hub VM as intermediary (relayed).

On AWS, the services used are identity access management (IAM), virtual private cloud (VPC) from the networking area, elastic compute cloud (EC2) from the compute area, simple storage service (S3) from the storage area and Braket from the quantum computing offerings. Additionally, the cloud database with the test results is an Aurora cluster on AWS.

For the IBM Cloud, the services used are Virtual Servers from the compute offerings, IBM Cloud Object Storage for the data lake and the IBM Quantum set (Composer, Lab and Qiskit SDK) from the quantum computing area.

From GCP, the services used are compute engine virtual machines, object storage and the IonQ Quantum machine provided by their partner IonQ. Moreover, the Sycamore23 was also tested.

The multi-cloud quantum system is a distributed platform that must be running on all the CSP at the moment of the data distribution. It is composed by a Linux server, specifically with Ubuntu 20.04 LTS, in each CSP listening on port 22. Each time it receives a communication request, it creates a client/server private connection with SSH and exchange the required files with the client. Once the connection is established, it starts receiving data packets from the client and send the data to the data lake passing through a quantum validation using the simulators or the quantum computer on the local CSP with the QKD protocol.

Quaternions Generation

The component to generate the complex data to be stored on files that will be split to test the multi-cloud distributed data lake and distributed quantum calculations was developed with the latest stable version of the Python programming language at the moment, which is 3.10.2. This component generates a number of files with quaternions represented by their associated matrices.

There are several modules, package and libraries to represent quaternions on Python. For example "quaternion", "pyquaternion" and "rowan". In this research, the "quaternion" module is used, as it has been confirmed, that it has better performance than the others [36], despite not fully supporting Euler angles.

Data Distribution

The data distribution component is in charge of distributing the files using the different API calls from each cloud storage data lake. This component is the client of the multi-cloud quantum system server.

QKD Authorization

The user authorization component is capable of authorizing a file transmission using QKD protocol to validate the credentials and accept the user data. This must be executed from the same node where the distribution component will run. It takes each local data file served from the distribution, then it will get the first 2 quaternion data bytes (16 bits) and transform them to an initial state on qubits. With this string of bits, it executes the QKD protocol with both simulators and real quantum machines, depending on the version used. It takes only 2 bytes (16 bits) due to the limitations on the current quantum simulators. For the quantum computers the limitation is lower, with 4 bits only. Based on this authorization phase, it decides if it will send the file to the data lake or not.

AWS

The AWS QKD component was developed in 3 versions. One with the Braket local simulator, one with the cloud simulator and other using real quantum computers.

The cloud simulator used was the SV1, which is a universal state vector simulator. For this paper, the state vector is the best choice, as the experiments didn't include added noise (density matrix or DM1) or graph needs (tensor network or TN1). Moreover, the SV1 is proven faster than the DM1 for circuits with less than 28 qubits.

In the following code snippet, the lines used to connect to the AWS QC (specifically the IonQ machine) and run one of the program (Alice-Eve interaction) are listed, as an example to show how the QC resources are utilized.

```
. . .
import boto3
from braket.aws import AwsDevice
from braket.circuits import Circuit
. . .
aws_account_id = boto3.client
("sts").get_caller_identity()["Account"]
my_bucket = "amazon-braket-afbfc6532108"
my_prefix = "simulation-output"
s3_folder = (my_bucket, my_prefix)
. . .
    # Set the quantum computer as device
    device = AwsDevice("
    arn:aws:braket:::device/qpu/ionq/ionQdevice")
    # run circuit
    m_shots = 1
    result = device.run(alice_eve_circuit,
    shots = m_shots).result()
. . .
```

There is shown an example of what we call a hybrid algorithm, as it perform actions in both classical machines (credentials revision) and quantum machines (circuit execution).

IBM

The IBM QKD component was developed in 2 versions. One with the Qiskit simulator and other using real quantum computers. During the initial tests, some limitations were detected on the available quantum computers, such as:

- Low number of available qubits
  Although the simulator can use up to 32 qubits, the available qubits on the real quantum computers (IBMQ) for public is 5. Then, the experiments were ran using 16 qubits with the simulator and 4 qubits with the IBMQ machine.
- Duration in queue
  During the initial tests, sometimes the duration in the IBMQ queue was for several hours. Therefore, the use of the simulator version is more efficient.

For this reason, some of the experiments were ran in both versions to compare them, but some were only ran using simulators to use more qubits, which is only offered by using the Qiskit simulator.

GCP

The GCP QKD component was developed in 2 versions. One with the Cirq local simulator and other using real quantum computers.

Eavesdropping Simulation

The eavesdropping simulation was made using an interception inside the hybrid algorithm. It is the same code as the execution without interception, but before the last mile VM receives the qubits, Eve will try to extract some information from them. As previously mentioned, an actual quantum network does not properly exists at the moment, and for this reason, the eavesdropping was simulated on the code directly.

Error Correction

The error correction method used was the repetition codes, similar to the one described in [37]. For this a circuit with equivalent number of codes and ansillas was configured in each CSP. Basically, the method consist in sending the same information several times and compare the results. With a high number of repetitions the solution of the calculation should be the value which is more repeated, and this will provide us with the answer of the transmitted data.

Data Lake

The distributed data lake was created using the CLI commands for each CSP. The create-bucket command on AWS, the bucket-create command on IBM and the mb command on GCP. The data files and the solution of the distributed quantum calculations will be stored there.

Results Gathering

We developed a program to gather the results. It runs on the hub machine, which will collect the results from each block uploaded and store it on a cloud database. This cloud database is located on AWS and based on Amazon Aurora.

Testing and Evaluation

In this section, we discuss the evaluation of the implemented solution.

Test Configurations

The evaluations were executed using the following configurations:

- Local machine
  The local machine was used to generate the data files for the experiments. Operating system: macOS Monterey 12.2.1, Processor: 2.3 GHz Dual-Core Intel Core i5, memory: 8 GB 2133 MHz LPDDR3.

Table 1 Quantum computers and simulators specifications

Item	Value
AWS QC	IonQ Aria device (gate based QPU)
IBM QC	IBM quantum system one
GCP QC	Google IonQ and Sycamore23
AWS simulator	SV1 and the Braket local simulator
IBM simulator	IBM Qiskit simulator
GCP simulator	Cirq simulator

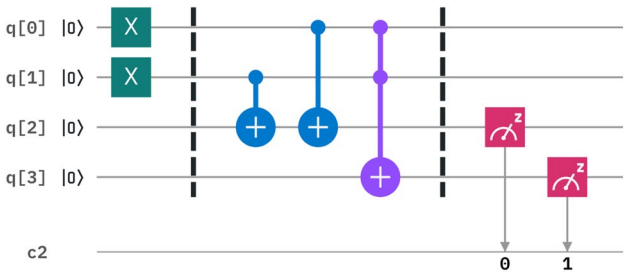

Fig. 6 Quantum adder circuit

- Cloud Hub VM and CSP Servers
  The cloud hub VM was used as the pivot to distribute the data files and as the communication bridge between the local machine and the other cloud components. The CSP servers are the machines that receives the data files and run the quantum experiments. Operating System: Ubuntu Linux 20.04 Focal Fossa, Processor: Intel(R) Xeon(R) 8259CL, memory: 8 GB. Note: The same Ubuntu version and memory size was used for all the 3 CSP.
- Quantum computers and simulators
  The quantum computers and simulators used in the experiment to run the QKD section of the hybrid codes are presented in Table 1.
- Data lake
  The data lake is the place in each CSP where the data resides after the QKD process and where the distributed quantum calculations solutions will be stored. AWS data lake: simple storage service (S3), IBM data lake: cloud object storage, GCP data lake: cloud storage.

All the duration times on the experiments were collected using the Linux time tool.

Percentage of Correct Distributed Calculations

This section shows the results of using the distributed quantum system to achieve a distributed simple math calculation. The idea is to use the built distributed quantum environment to perform a distributed calculation. In the experiments, each quantum machine calculated a sum and then, the results were obtained from the hub VM and multiplied. Finally, a program checked how many results were correctly calculated.

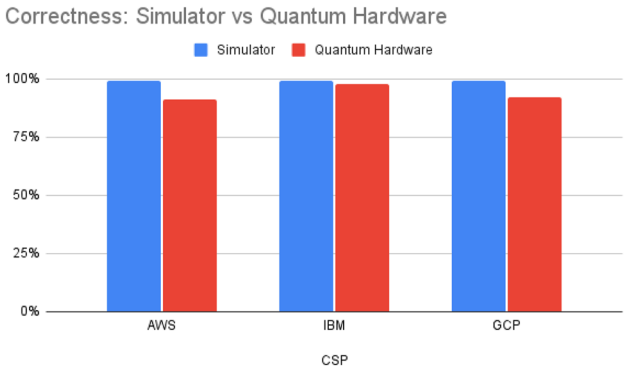

Fig. 7 Percentage of correctness: simulator vs quantum hardware

The experiment executed 1000 simple math calculation between two single digit binary number on the quantum machines (using the adder circuit [38] shown in Fig. 6). The results were 88% correct and 12% incorrect.

These tests confirms that the distributed calculations using heterogeneous quantum machines from different CSPs is possible.

Percentage of Correct Distributed Calculations Using Error Correction Methods

This section shows the results of using the distributed quantum system to achieve a distributed simple math calculation while using basic error correction methods such as repetition. For this specific experiment, the use of simulators to perform the error correction sections were crucial as they demand a major number of qubits for the proposed operations.

The experiment executed 1000 simple math calculation between two single digit binary number on the quantum machines using the adder circuit previously mentioned. The results were 96.8% correct and 3.2% incorrect. It is noticeable how the error correction methods works fine in a multi-cloud quantum approach as well. Other error correction methods such as Shor codes are in testing process at the moment in the same distributed quantum infrastructure. The results will be published in future works.

These tests confirms that the error correction methods increase the correctness of the results and can be used in multi-cloud quantum architectures as well.

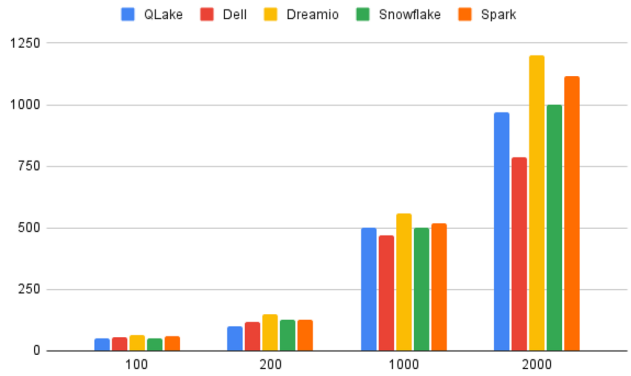

Fig. 8 Comparison between QLake and existing classical alternatives

Percentage of Correct Authorizations

These experiments show what percentage of the executions between Alice and Bob were successful and what percentage failed due to quantum machine noise.

As shown in Fig. 7, both the percentage of correctness results were achieved when using the simulators and when using quantum hardware with each vendor.

The experiment was ran one hundred times with 16 qubits when using simulators and 4 qubits when using quantum hardware. This should provide less advantage to the simulators. However, as they are noise free, they resulted in correct outputs most of the time (99% as per the experiments). Meanwhile, the results on hardware were 91% on AWS Braket, 98% on IBM Q and 92% on GCP.

Comparison Between Existing Distributed Data Lakes and Multi-cloud QLake

For this set of experiments, the following existing multi-cloud alternatives were used to compare both the performance and the upload duration:

- Dell Apex multi-cloud data services
  This option was tested with help of the hardware team on Dell Ireland, because the actual service request a minimum of 26 TB to be allocated. The experiments were ran with a demo version of the service with minimal storage (up to 10 GB). It works with a distributed data lake using any combination from AWS, Azure, GCP and Oracle Cloud. In the experiments, the AWS and GCP were used. One of the advantages of this product is the private connection from Dell to all the partner CSPs. This improve the data file upload speed.
- Dremio multi-cloud
  For this experiment, the standard (free) version was used. The CSP used were AWS and Azure, as those are more documented on this tool website. Some benefits of using this tool are the tool is free and also it is easy to use.
- SnowFlake
  Snowflake is one of the most used multi-cloud data lake. This is a pay-as-you-go service and it supports AWS, Azure and GCP. For this experiment, AWS and GCP were used.
- Terraform + Apache Spark This option prepare an environment using AWS and GCP as CSPs and Apache Spark to upload the data.

In all cases only two CSPs were used because of the limitation in some of the products to support other CSPs. Therefore, for this section, the tool of this research was modified to use only two of the CSPs as well. The chosen CSPs were AWS and GCP. All the experiments used the same source machine and the same cloud servers.

In Fig. 8, the duration time comparison for these 4 alternatives and the implemented tool on this project were shown. The experiment uploaded 100, 200, 1000 and 2000 quaternion files using each product/tool.

As shown in Fig. 8, the QLake tool using quantum simulators was the best with less amount of files, however with large number of files, the Dell solution was the best. This means that the quantum alternative works better than existing alternatives, except just one. With some improvements, the quantum alternative could approach further the duration times of the best option and reduce it in the future.

Discussion and Conclusions

Based on our experimentations, the overall QLake system is working as expected, and the output of the program is consistent as planned. The main conclusion of this research is the confirmation of the possibility to distribute quantum computation between multiple CSPs, which when adding a basic error correction method, such as repetition or bit flip codes, the correctness increases. Therefore, if we use a more complex error correction mechanism, the accuracy will increase even more. Also, the results shown that the provision of the same multi-cloud architecture can be used to compute different quantum calculations using several qubits in each CSP and retrieving the results to later execute local operations over the merged results.

However, with the current quantum computers there are a good number of limitations that leads to the current impossibility to execute big workloads on these. For experimentation, the use of quantum simulators is a good option, but although they use real quantum procedures and theory, in the backend it uses classical machines, which affects the fact of using real quantum technologies.

Future Perspectives

Perhaps the current quantum computers have limitations, but there are extensions that can be developed over this project. Firstly, the use of quantum computers from other CSPs, such as the ones developed by D-wave systems, Honeywell or Alibaba. Also, by continuously experiment with new quantum machines that are delivered each year to reach new best upload times, and once a quantum global network is created, test the complete QKD section without a simulation, but with a proper measurement.

Another future beneficial extension to this work would be a completion of tests and experiments including more complex error correction methods, such as low density parity check codes with a high encoding rates, to reduce the errors while still showing excellent performance. Since the main point to demonstrate in this research is related to the possibility of distributing quantum calculations in a multi-cloud architecture and not the error rates methods.

As a result, for testing purposes and to extend the quantum hybrid and cloud development community, this project code can be used to study further considerations. Of course, the need of more powerful quantum computers with more qubits would be crucial to achieve better results, improved performance and new areas of research. Hopefully, in the near future, these quantum machines with more compute power will be available.

Author Contributions Not applicable

Funding Open Access funding provided by the IReL Consortium. Not applicable

Availability of Data and Materials Not applicable

Code availability Not applicable

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

Ethics approval Not applicable

Consent to participate Not applicable

Consent for publication Not applicable

References

1. Cumming R, Thomas T. Using a quantum computer to solve a real-world problem—what can be achieved today? (2022).
2. Arute F. Quantum supremacy using a programmable superconducting processor. Nature. 2019;574:505–10.
3. Wang Y. 16-qubit ibm universal quantum computer can be fully entangled. Quantum Inf. 2018;4:46.
4. Nielsen MA, Chuang IL. Quantum computation and quantum information: 10th anniversary. Cambridge: Cambridge University Press; 2010. https://doi.org/10.1017/CBO9780511976667.
5. Kim Y, Eddins A, Anand S. Evidence for the utility of quantum computing before fault tolerance. Nature. 2023;618:500–5. https://doi.org/10.1038/s41586-023-06096-3.
6. Berg E, Minev ZK, Kandala A. Probabilistic error cancellation with sparse Pauli–Lindblad models on noisy quantum processors. Nat Phys. 2023;16:1116–21. https://doi.org/10.1038/s41567-023-02042-2.
7. O'Reilly Survey: Cloud Adoption 2021. https://www.oreilly.com/radar/the-cloud-in-2021-adoption-continues/. Accessed 1 Mar 2022 (2022).
8. Cloud Security Alliance Survey. https://cloudsecurityalliance.org/artifacts/cloud-security-and-technology-maturity-survey/. Accessed 1 Mar 2022 (2022)
9. Benioff P. The computer as a physical system: a microscopic quantum mechanical hamiltonian model of computers as represented by turing machines. J Stat Phys. 1980;22:563–91.
10. Feynman R. Simulating physics with computers. Int J Theor Phys. 1982;21:467–88.
11. Shor PW. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM J Comput. 1997;26(5):1484–509. https://doi.org/10.1137/s0097539795293172.
12. IDC Forecasts Worldwide Quantum Computing Market to Grow to \$8.6 Billion in 2027. https://www.idc.com/getdoc.jsp?containerId=prUS48 414121. Accessed 5 Apr 2022 (2022).
13. Dirac PAM. A new notation for quantum mechanics. Math Proc Camb Philos Soc. 1939;35(3):416–8. https://doi.org/10.1017/S0305004100021162.
14. Bloch F. Nuclear induction. Phys Rev. 1946;70:460–74. https://doi.org/10.1103/PhysRev.70.460.
15. Beckers A, Tajalli A, Sallese J-M. A review on quantum computing: qubits, cryogenic electronics and cryogenic MOSFET physics.
16. Bennett CH, Brassard G. Quantum cryptography: public key distribution and coin tossing. Theoret Comput Sci. 2014;560:7–11. https://doi.org/10.1016/j.tcs.2014.05.025. (**Theoretical Aspects of Quantum Cryptography—celebrating 30 years of BB84**).
17. Kaiiali M, Sezer S, Khalid A. Cloud computing in the quantum era. In: 2019 IEEE conference on communications and network security (CNS), pp 1–4 (2019). https://doi.org/10.1109/CNS44998.2019.8952589.
18. Ravi GS, Smith KN, Gokhale P, Chong FT. Quantum computing in the cloud: analyzing job and machine characteristics (2022).
19. Hussain M, Wei L-F, Abbas F, Rehman A, Ali M, Lakhan A. A multi-objective quantum-inspired genetic algorithm for workflow healthcare application scheduling with hard and soft deadline

constraints in hybrid clouds. Appl Soft Comput. 2022;128: 109440. https://doi.org/10.1016/j.asoc.2022.109440.
20. Zhang M, Fu Y, Wang J, Lai J. Research on task scheduling scheme for quantum computing cloud platform. In: Proceedings of the 2022 6th international conference on cloud and big data computing. ICCBDC '22, pp. 7–11. Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3555962.3555964.
21. Liu L, Dou X. Qucloud: a new qubit mapping mechanism for multi-programming quantum computing in cloud environment. In: 2021 IEEE international symposium on high-performance computer architecture (HPCA), pp 167–178 (2021). https://doi.org/10.1109/HPCA51647.2021.00024.
22. Pozzi MG, Herbert SJ, Sengupta A, Mullins RD. Using reinforcement learning to perform qubit routing in quantum compilers. ACM Trans Quantum Comput. 2022;3:2. https://doi.org/10.1145/3520434.
23. Smith KN, Ravi GS, Murali P, Baker JM, Earnest N, Javadi-Cabhari A, Chong FT. Timestitch: exploiting slack to mitigate decoherence in quantum circuits. ACM Trans Quantum Comput. 2022;4:1. https://doi.org/10.1145/3548778.
24. Iten R, Moyard R, Metger T, Sutter D, Woerner S. Exact and practical pattern matching for quantum circuit optimization. ACM Trans Quantum Comput. 2022;3:1. https://doi.org/10.1145/3498325.
25. Nguyen T, Lyakh D, Dumitrescu E, Clark D, Larkin J, McCaskey A. Tensor network quantum virtual machine for simulating quantum circuits at exascale. ACM Trans Quantum Comput. 2022;4:1. https://doi.org/10.1145/3547334.
26. Zeydan E, Baranda J, Mangues-Bafalluy J. Post-quantum blockchain-based secure service orchestration in multi-cloud networks. IEEE Access. 2022;10:129520–30. https://doi.org/10.1109/ACCESS.2022.3228823.
27. Sudhakar Reddy N, Padmalatha VL, Sujith AVLN. A novel hybrid quantum protocol to enhance secured dual party computation over cloud networks. In: 2018 IEEE 8th international advance computing conference (IACC), pp 142–149 (2018). https://doi.org/10.1109/IADCC.2018.8692128.
28. Zhao B, Zha X, Chen Z, Shi R, Wang D, Peng T, Yan L. Performance analysis of quantum key distribution technology for power business. Appl Sci. 2020;10:8. https://doi.org/10.3390/app10082906.
29. Srivastava V, Pathak RK, Kumar A, Prakash S. Using a blend of brassard and benett 84 elliptic curve digital signature for secure cloud data communication. In: 2020 international conference on electronics and sustainable communication systems (ICESC), pp. 738–743 (2020). https://doi.org/10.1109/ICESC48915.2020.9155663.
30. Liu R, Rozenman GG, Kundu NK, Chandra D, De D. Towards the industrialisation of quantum key distribution in communication networks: a short survey. IET Quantum Commun. 2022;3:151–63. https://doi.org/10.1049/qtc2.12044.
31. Wootton JR, Loss D. Repetition code of 15 qubits. Phys Rev A. 2017;97: 052313.
32. Calderbank AR, Shor PW. Good quantum error-correcting codes exist. Phys Rev A. 1996;54(2):1098–105. https://doi.org/10.1103/physreva.54.1098.
33. Hamilton WR. On quaternions or on a new system of imaginaries in algebra. Philosophical Magazine. Trinity College Dublin (1844–1850).
34. GitHub platform—Jose Lo Huang. https://github.com/artneutro/MTU. Accessed 11 Mar 2022 (2022).
35. The European Quantum Communication Infrastructure (EuroQCI) Initiative. https://digital-strategy.ec.europa.eu/en/policies/european-quantum-communication-infrastructure-euroqci. Accessed 6 Nov 2023 (2023).
36. Ramasubramani V, Glotzer SC. rowan: a python package for working with quaternions. J Open Source Softw. 2018;3(27):787. https://doi.org/10.21105/joss.00787.
37. Wootton JR. Benchmarking near-term devices with quantum error correction. Quantum Sci Technol. 2020;5(4): 044004. https://doi.org/10.1088/2058-9565/aba038.
38. Qiskit—The Atoms of Computation. https://qiskit.org/textbook/ch-states/atoms-computation.html. Accessed 12 Apr 2022 (2022).