Received 7 November 2023, accepted 4 December 2023, date of publication 13 December 2023,
date of current version 22 December 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3342749

RESEARCH ARTICLE

Toward a Quantum-Science Gateway: A Hybrid Reference Architecture Facilitating Quantum Computing Capabilities for Cloud Utilization

ATTILA CSABA MAROSI[1], **ATTILA FARKAS**[1,2], **TAMÁS MÁRAY**[1], **AND RÓBERT LOVAS**[1,3]
[1]Laboratory of Parallel and Distributed Systems, Institute for Computer Science and Control (SZTAKI), Hungarian Research Network (HUN-REN), H-1111 Budapest, Hungary
[2]Doctoral School of Applied Informatics and Applied Mathematics, Óbuda University, H-1034 Budapest, Hungary
[3]Institute for Cyber-Physical Systems, John von Neumann Faculty of Informatics, Óbuda University, H-1034 Budapest, Hungary

Corresponding author: Attila Csaba Marosi (attila.marosi@sztaki.hun-ren.hu)

This work was supported in part by the European Union within the framework of the Artificial Intelligence National Laboratory under Project RRF-2.3.1-21-2022-00004, in part by the Ministry of Innovation and Technology National Research, Development and Innovation Office (NRDI) Office through the 'Research on Cooperative Production and Logistics Systems to Support a Competitive and Sustainable Economy' under Grant TKP2021-NKTA-01 NRDIO, and in part by Hungarian Scientific Research Fund (''Országos Tudományos Kutatási Alapprogramok'') under Grant K 132838. The work of Róbert Lovas was supported in part by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences; and in part by the UNKP-22-5 New National Excellence Program of the Ministry for Innovation and Technology, funded by the National Research, Development and Innovation Fund (Bolyai+).

ABSTRACT The emerging availability of quantum compute resources fosters the examination of its exploitation possibilities in different scientific domains, like artificial intelligence, manufacturing or finance. A significant number of research scientists primarily rely on cloud computing infrastructures while conducting research, whereas access to real (mostly remote) quantum hardware resources requires the deployment and proper configuration of different software components. In this paper we present a hybrid, cloud-based reference architecture that lowers the entry barrier to start new experiments with a wide range of quantum compute resources. The solution facilitates the execution of distributed quantum computing simulations in traditional cloud environments, and also access to several remote quantum compute resources. The reference architecture is highly portable to various cloud platforms, resulting in efficient adaptation and application possibilities by research communities. The paper describes our related experiences using commercial cloud providers and on the federated, OpenStack-based research infrastructure of the Hungarian Research Network (abbreviated as HUN-REN).

INDEX TERMS Cloud computing, quantum computing, reference architecture, simulation, science gateway.

I. INTRODUCTION

As near-term quantum computers are becoming widely available for end users as well [2], [3], [4], [5], [6], there is an obvious requirement to simplify their usage for interested researchers as much as possible by lowering the entry barrier. Quantum computing resources are still complicated to use with their user interfaces, algorithms and applications are far from being at the level of abstraction where they can be easily used by researchers from other fields than quantum experts. Usually accessing these resources requires the deployment of various development frameworks, which contain all the necessary tools [7], [8]. This might be a straightforward process, however with the increasing number of such frameworks, the task of interested researchers also increases: the installation and configuration documentation of the different frameworks must be examined carefully, and also, the deployment steps must be performed in some sort of environment, which can be either the local machine of the user, a cloud-based virtual machine service, or a bare metal service. Additionally, once the development framework is installed, further tools or environments are needed through which the functionalities of the selected quantum framework can be examined.

The associate editor coordinating the review of this manuscript and approving it for publication was Fabrizio Messina.

There is therefore a clear gap between quantum computing as a tool and its potential real users. This is a barrier to the uptake of quantum computing and also a setback for the development of quantum computing as a technology, since as long as the number of users and the real application results are still limited, the development of the technology cannot proceed at a faster pace.

Our goal is to build a gateway that significantly lowers the barrier to entry for scientists wishing to learn and apply quantum computing by making quantum resources, algorithms and applications more easily and simply available, starting from a known environment. In general, *science gateways* are web-based platforms that provide researchers with user-friendly access to computational resources, data, and scientific tools. They streamline complex processes, promote collaboration, and democratize access to computing resources, facilitating scientific research, and accelerating discoveries.

In this paper we present a hybrid reference architecture for bringing quantum computation closer to the research communities as the targeted end users. It incorporates major quantum software development kits (SDKs), including support for machine learning, and access to different quantum devices. We also provide a set of examples and present preliminary benchmark results. These can serve as practical guides for constructing solutions to other (predefined) problems. Beyond these, our reference architecture aims to deliver a user-friendly interface for researchers.

The structure of the paper is as follows: in Section II we present relevant fields of cloud computing, quantum resources, and we introduce the reference architecture concept. In Section III, we present the our reference architecture in detail. Following that, in section IV we examine different deployment options of our reference architecture using diverse hosting environments. In Section V we discuss future improvement plans, and finally, Section VI presents the paper's conclusions.

II. STATE OF THE ART

Accessing quantum computing resources via the consumption models introduced by commercial clouds provide a feasible solution for making quantum computing widely accessible [9]. Thus, Quantum Cloud Computing (QCC) or Quantum Computing as a Service (QCaaS) [10] enables running tasks without the need of owning and operating physical hardware [11], [12]. Furthermore, applying the best practices and the existing knowledge encapsulated in cloud reference architectures could allow the creation of new best-of-breed solutions that harness the power of both cloud and quantum computing. Quantum computing, cloud computing and reference architectures are all rapidly evolving fields with wide range of research literature available. However, to keep related works focused, in this section we are investigating them only in three narrow categories with close connection to our work. First, we discuss (i) reference architectures in general. Next, we focus on (ii) how they are utilized in cloud computing. Finally, we discuss (iii) quantum resources in terms of available cloud-enabled offerings to be incorporated into a reference architecture. We note here, that we relate the presented related work concepts to our solution in Section III.

A. REFERENCE ARCHITECTURES AND CLOUD COMPUTING

Architecture blueprints in software architectures allow the reuse of existing knowledge and best practices when creating new solutions. There are different definitions proposed, e.g., [13], [14], [15], and [16] with the main ideas of (i) promoting re-usability, (ii) incorporating best practices, (iii) using high or low abstraction levels, and (iv) serving certain use cases. In general high abstraction level architectures typically contain approaches and system design principles, but lack concrete implementation references. Contrary, low-level architectures focus on the implementation details, in cloud computing typically using platform and software based services of a particular cloud provider. These type of services are referred as Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS) services, respectively. Additionally, such blueprints can utilize open-source software components as well.

The concept of reference architectures [17], [18], [19] enables that a well-tested, proven architecture can be published for reproducing some required functionality. It allows architects to compose a service stack of cooperating entities in a way that the verified composition can be deployed by other users easily, without the need to configure the entities of the stack, or to deploy the entities individually.

To understand reference architectures in cloud computing we need to introduce the two dimensions on which cloud services are offered [20]: (i) service and (ii) deployment models.

The NIST definition for Cloud Computing [20] defines three service models for cloud computing. These service models are layered using a top-down view as follows: (i) in *Software as a Service (SaaS)* applications (e.g., web-based email) are provided to the user and managed by the provider on its infrastructure; (ii) *Platform as a Service (PaaS)* provides the capability to deploy applications and services on to the cloud without the need for the consumer to provision and manage the hosting resources; and finally (iii) *Infrastructure as a Service (IaaS)* provides the capability for the consumer to provision compute, storage, network, etc. resources without the responsibility for the underlying cloud infrastructure.

Additionally, The NIST Definition for Cloud Computing [20] defines four deployment models for cloud computing, where the first three characterise the accessibility and the fourth targets the composition of the cloud. First, (a) in a *private cloud* the infrastructure is provisioned for the use by a single organisation (that may be hosting different consumers, such as business units or departments). Private cloud resources are mostly implemented using an open-source cloud software, like OpenStack [21] or OpenNebula [22];

(b) *public clouds* are publicly accessible and infrastructure is provisioned for open use following typically a pay-as-you-go or a subscription model. Most well-known public cloud providers are Amazon Web Services [23], Microsoft Azure [24], [25], Google Cloud Platform [26] and IBM Cloud [27]; (c) *community clouds* are provisioned for the exclusive use by a larger but targeted community of users typically with a shared purpose and are a blend of public and private clouds; and (d) in *hybrid clouds* the infrastructure is composed of two or more distinct cloud infrastructures (public, private or community).

Public cloud providers offer diverse set of reference architectures. These are all low-level architectures aimed at commercializing their cloud-based services. For example Microsoft Azure currently offers 118 architecture blueprints for data analytics [28], and AWS offers 64 reference architectures for big data and data analytics [29]. On the contrary, the "Industrial Internet Reference Architecture" [30] and "Reference Architectural Model Industry 4.0" [31] represent the joint effort of several organizations and companies, however, they are high-level concepts.

HUN-REN Cloud [1] in Hungary is a federated, community cloud of the Hungarian Research Network (HUN-REN) that offers low-level reference architectures utilizing open source components to the science communities. The goal of HUN-REN Cloud is supporting Hungarian scientists with elastic, virtualized computing resources. The low-level reference architectures offered by HUN-REN Cloud include amongst others blueprints for Horovod [32], Apache Spark [33], Jupyter development environments, and Kubernetes. For example the researchers can use these reference architectures to easily set up a Kubernetes cluster in HUN-REN Cloud, without the need to know the details of deployment or internal specifics. The reference architectures always come with documentation on deployment and usage. They are utilizing open source orchestration tools (such as Terraform [34]) and configuration management tools (such as Ansible [35]) for deployment.

This science cloud provides a combined 5904 vCPUs, with 28TB RAM, 1248TB HDD and 338TB SSD storage. It also offers a total of 68 GPUs with 2400GB RAM, and 584TFlops double precision, 1174TFlops single precision and 13768TFlops FP16 Tensor performance. The presented platform is deployed in the site of the Institute of Computer Science and Control (abbreviated as SZTAKI) of the federated cloud. This site contains (among others) 10 compute nodes, each with 2x Intel Xeon Gold 6230R 26-Core CPUs and 768GB of RAM. Additionally, there are 6 HDD and 6 SSD storage nodes, each with 192TB and 92TB of raw storage, respectively. The cloud is based on OpenStack Wallaby, and the storage backend is built using Ceph Pacific. All virtual machines have 10Gbps networking by default, with optional configuration settings for 30-35Gbps. HUN-REN Cloud is connected to the Hungarian academic backbone with 100Gbs network.

B. QUANTUM RESOURCES

The purpose of the works presented is the emerging availability of quantum compute resources. Just like cloud systems, quantum resources can also be divided into different categories, based on for example the type of computing model supported, the technology used to implement qubits, or the availability of the resources.

In [36] an overview of quantum computing literature is provided, accompanied by the introduction of a taxonomy for quantum computing. This taxonomy is employed to categorize and analyze numerous related studies and to highlight research areas for further investigation. The authors present the current state-of-the-art in quantum computing through an exploration of quantum software tools and technologies, post-quantum cryptography, and advancements in quantum computer hardware development. Additionally, the article discusses various open challenges for future research in quantum computing.

Next, we focus on discussing the offerings of hardware providers following the two main quantum computing models: the *quantum annealing model* [37], [38], and the *quantum circuit model* [39], [40], [41]. A summary of the quantum resources discussed in the following paragraphs can be found in Table 1 and Table 2. Similarly, a summary of the SDKs and environments discussed in the following paragraphs can be found in Table 3. Additionally, an in-depth discussion and comparison of quantum programming languages can be found in [42].

The quantum annealing model is an optimization-oriented approach, that can be used to find minimum values for an objective function representing a problem. Thus, in case of the annealing model, the user must formulate the problem as an objective function, where the minimum value of the objective function represents the best solution for the problem. The task of the quantum hardware is to map the properly defined function into a physical system, run the system following the rules of quantum mechanics, and take samples of the system.

There is basically one major company offering quantum computers following the *quantum annealing model*: D-Wave Systems. They have a number of quantum processing unit (QPU) generations, with number of qubits ranging from 2000 up to 7000 [43]. The problems to be solved by the D-Wave hardware must be represented as an objective function, in some sort of quadratic model. D-Wave QPUs support binary, discrete and constrained quadratic models. The areas of applications supported by the D-Wave systems ranges from simple optimization problems up to machine learning applications.

The big advantage of D-Wave systems is that they do not only offer access to their QPUs, but also provide a hybrid system, called Leap, in order to ease the solving of more complex problems, using hybrid computation of classical and quantum resources as well. Access to D-Wave resources is available after registration for a one-month evaluation period, under given quota restrictions: the amount of QPU and hybrid execution time is limited. The one-month period can be extended, given that an open-source project is provided.

The development framework of applications for D-Wave resources is called Ocean SDK [44]. This is a Python-based SDK, offering a very diverse set of functionalities for defining problems for the D-Wave hardware. It also provides solvers and samplers for processing the defined problems locally, on a QPU, or in the cloud infrastructure of Leap, using both QPUs and classical resources in a hybrid manner. Besides the Ocean SDK, third-party IDEs are generally supported both locally and cloud-based. The IDEs have to implement the described Development Containers specification. D-Wave recommends GitHub Codespaces or local IDE like VS Code. This support replaces the previously available Leap IDE.

On the other hand, the *quantum circuit model* is closer to the classical computation: it offers a set of quantum gates [41] for performing operations on qubits, but in contrary to classical systems, where a bit can have values 0 and 1, a qubit can have the values $|0\rangle$, $|1\rangle$, or any other state in the superposition of these.

The vast majority of quantum hardware providers implement the quantum circuit model with Superconducting architecture, for example, Google Quantum AI [45] IBM Quantum [46], IQM [47], OxfordQuantumCircuits [48], and Rigetti [49], [50]. Some providers use a Trapped ion architecture like AQT [51], IonQ [4] and Quantinuum [52]. Lastly, other hardware providers use an Analog quantum approach with Neutral atoms architecture, like Pasqal [53] and QuEra [54], [55]. The summary of these providers with their latest QPU capability and supported SDKs is presented in Table 1. Access to the hardware of these providers is possible either directly, or using a public cloud provider offering third-party quantum computation services, like Amazon Braket [56].

Amazon Braket [56] works as proxy provider for accessing QPU devices of IonQ, Rigetti or Oxford Quantum Circuits, among others. Development is available with the help of the Amazon Braket SDK. Amazon Web Services (AWS) also provides an interactive development environment similar to IBM Quantum Labs through its managed SageMaker notebook service [57]. AWS has a number of simulators available, with different properties, and different pricing as well. Usage of the simulators is free in the free tier for a given amount of time, any other usage is billed against the user. Usage of QPU devices is also billed.

Azure Quantum provides a service similar to Amazon Braket: it offers access to QPU devices through a public cloud provider interface. The set of hardware providers includes IonQ, Rigetti and Pasqal among others. Resources offered by Azure Quantum can be accessed from Cirq, Qiskit, and the Azure Quantum Development Kit. The latter one also offers the high-level Q# programming language, for developing quantum programs.

Google doesn't offer its QPU devices publicly available [45] but provides the Cirq [58] framework for creating quantum circuits. However, access to other hardware providers (like IonQ, Rigetti, Pasqal and Azure Quantum) is available through plugins.

IBM offers a complete solution for quantum computation [59], [60]. They not only offer the Qiskit SDK [61] for creating quantum circuits, but also provide cloud-based simulator services, and access to QPU devices. The IBM Quantum ecosystem also provides a graphical tool for designing circuits, called IBM Quantum Composer [62]. Additionally, IBM Quantum Platform offers an interactive development environment for creating Python applications using the functionalities of Qiskit [63]. Registered users have access to 5 and 7-qubit QPUs free of charge for 10 minutes per month besides the cloud-based simulators. Following the pay-as-you-go scheme, access to 27 qubit devices is also possible. More recent QPUs, offering 127 qubits are available after negotiation.

"Bare metal" quantum hardware providers, like IonQ or Rigetti usually provide their own entry point to gain access to their resources offered. They also may provide a custom SDK for developing applications for their hardware but mostly tend to make their resources available through public cloud providers as well (like Amazon Braket or Azure Quantum). The summary of the Quantum Cloud providers with the supported QPCs, SDKs and Free Tiers plans are summarised in Table 2.

Important new opportunities for researchers wishing to use quantum computers will soon be opened up with the addition of 6 new quantum computers to EuroHPC's research oriented supercomputing infrastructure. The EuroHPC JU (European High Performance Computing Joint Undertaking) has 34 participating countries and the EU represented by the European Commission. Currently there are 7 operational TOP500 grade EuroHPC supercomputers (Vega, Karolina, Discoverer, Meluxina, LUMI, Leonardo, Deucalion) with 3 systems underway (MareNostrum, Jupiter, Daedalus) and more to come. Lumi is ranked third and Leonardo fourth in the TOP500 (June 2022). Jupiter will be the first European exascale supercomputer and will be installed in Germany. This powerful infrastructure will soon be complemented by six new quantum computers with different architectures, opening up new opportunities for a wide range of researchers. These system will not only be integrated with traditional supercomputers, but according to EuroHPC the resources will be made globally available to the users by all possible means including cloud based access, for which different research and innovation initiatives are needed.

In summary, we can observe approaches at both higher and lower levels for providing access to quantum computing resources. At the higher level, abstract methods such as iQuantum [10] aim to offer a system model for hybrid quantum computing environments, including brokering. However, these high level approaches typically focus on simulation and modeling. In contrast, at the lower level, there are numerous vendor-specific quantum computing libraries and meta-libraries (e.g., Qiskit, Amazon Braket [56],

TABLE 1. Quantum resources: quantum processing unit providers.

Provider	Type	Architecture	Latest QPU	No. of Qubits	Release Date	SDK Support	Ref. Arch. Support
D-Wave	Quantum annealing	Superconducting	Advantage	5000+	2021	Ocean SDK	Yes
IBM Quantum	Circuit-based	Superconducting	Osprey	433	2022 Nov	Qiskit, CQ tket, QCW Forge, ProjectQ, PennyLane	Yes
IQM	Circuit-based	Superconducting	N/A	20	2023 Oct	Qiskit, CQ tket, ProjectQ, PennyLane,TensorFlow Quantum	Yes
OxfordQuantum Circuits	Circuit-based	Superconducting	Lucy	8	2022 Feb	Amazon Braket SDK, Qiskit, CQ tket	Yes
Rigetti	Circuit-based	Superconducting	ASPEN-M-3	80	2022 Dec	PyQuil, Cirq, Qiskit, CQ tket, Q#, QCW Forge, ProjectQ, PennyLane, TensorFlow Quantum	Yes
AQT	Circuit-based	Trapped ion	PINE system	24	2021 June	Qiskit, Cirq, CQ tket, ProjectQ, PennyLane, TensorFlow Quantum	Yes
IonQ	Circuit-based	Trapped ion	Forte	32	2022	Qiskit, Cirq, CQ tket, Q#, QCW Forge, ProjectQ, PennyLane, TensorFlow Quantum	Yes
Quantinuum	Circuit-based	Trapped ion	H2	32	2023 May	Qiskit, Cirq, CQ tket, Q#, PennyLane, TensorFlow Quantum	Yes
Pasqal	Analog quantum	Neutral atoms	Gen1	100	2022 May	Cirq, PennyLane, Pulser, TensorFlow Quantum	Yes
QuEra	Analog quantum	Neutral atoms	Aquila	256	2022 Nov	Amazon Braket SDK, Qiskit, CQ tket	Yes

TABLE 2. Quantum resources: cloud service providers.

Cloud service	QPC Support	SDK Support	Free Tier	Ref. Arch. Support
Amazon Braket	OQC, Rigetti, IonQ, QuEra	Amazon Braket SDK, Qiskit, PennyLane	1 hour free simulation time / month	Yes
Azure Quantum	Rigetti, IonQ, Quantinuum	Qiskit, Cirq, Q#	$500 USD free credit for 6 months	Yes
QC Ware	IBM Quantum, Rigetti, IonQ	QC Ware Forge	Free 1 min QCT	No
D-Wave Leap	D-Wave Advantage	Ocean SDK	Free 1 min QPU access per month	Yes
IBM Quantum Platform	IBM Quantum	Qiskit, CQ tket, QCW Forge, ProjectQ, PennyLane	Free 10 mins QPU access per month	Yes
PASQAL Cloud Services	Pasqal	Cirq, PennyLane, Pulser, TensorFlow Quantum	N/A	Yes
QuEra Cloud	QuEra	Amazon Braket SDK, Qiskit, PennyLane	Amazon Free Tier	Yes

Cirq, PennyLane) that strive to provide a unified interface for accessing various quantum hardware and/or service providers. Nevertheless, these support only a subset of providers and resources, with a limited range of predefined use cases. Our reference architecture seeks to provide a middle ground. It incorporates major quantum software development kits (SDKs) and frameworks, such as those provided by IBM [46], D-Wave [43], IonQ [4], [5], and quantum devices available through Amazon Braket [56] (e.g., IonQ or Rigetti [49], [50] QPU devices). The architecture also contains frameworks enabling experiments with quantum resources in the field of machine learning: Qiskit Machine Learning [61] and PennyLane. We also provide a set of examples and present preliminary benchmark results; these can serve as practical guides for constructing solutions to other (predefined) problems. On top of these, our reference architecture aims to provide a user-friendly interface for researchers. Furthermore, by relying on existing reference architectures, such as those for Kubernetes or Apache Spark, our solution encompasses the best practices from these reference architectures and is cloud vendor-agnostic as well.

III. REFERENCE ARCHITECTURE DESIGN

In this section the design of the quantum reference architecture is presented. The discussion includes the requirements based on which the work was started, the components that have been selected to participate in the reference architecture, and the implementation as well.

The motivation for the development of the reference architecture was the increasing availability of quantum computing resources. As it was presented in the previous section, a number of QPU device implementations are currently available for experimenting with. Thus, it was natural to investigate if they could be useful for scientific domains represented by the users of HUN-REN Cloud.

We started to experiment with D-Wave resources, as one could easily gain access to its most recent resources, free of charge within a given resource usage quota. It turned out that the Ocean SDK offered by D-Wave is relatively easy to use, and there were a number of examples already provided, proving the usability of their quantum devices in a number of application areas. Based on the provided examples, we have selected a few and started to examine them in detail, based on which we have created some examples on our own as well.

TABLE 3. Quantum computing tooling: SDKs and environments.

Tool Name	Type	Supported Programming Languages	Description	Supported Providers	License	Ref. Arch. Support
Qiskit	SDK	Python	"An open-source SDK for working with quantum computers at the level of pulses, circuits, and algorithms."	IBM Quantum, IQM, OQC, Rigetti, AQT, IonQ, Quantinuum, QuEra	Apache 2.0	Yes
IBM Quantum Learning	Learning Environment	Python (JupyterLab)	Hosted environment for experimenting with quantum computing resources.	IBM Quantum	Proprietary	No
Amazon Braket SDK	SDK	Python	"A Python SDK for interacting with quantum devices on Amazon Braket."	Amazon Braket (OQC, Rigetti, QuEra and IonQ)	Apache 2.0	Yes
Cirq (Google)	SDK	Python	"A python framework for creating, editing, and invoking Noisy Intermediate Scale Quantum (NISQ) circuits."	Rigetti, AQT, IonQ, Quantinuum, Pasqal	Apache 2.0	Yes
Azure Quantum Development Kit	SDK	Q#, Python (qdk-python)	The required SDK to interface with the Azure Quantum service.	Rigetti, IonQ, Quantinuum	MIT	No
pyQuil	SDK	Python	Build and execute Quil programs using Python	Rigetti Quantum Cloud Services	Apache 2.0	Planned
Ocean SDK	SDK	Python	"Ocean is D-Wave's suite of tools for solving hard problems with quantum computers."	D-Wave	Apache 2.0	Yes
CQ tket	SDK	Python	"pytket is a python module for interfacing with tket, a quantum computing toolkit and optimising compiler developed by Quantinuum."	IBM Quantum, IQM, Rigetti, AQT, IonQ, Quantinuum, Amazon Braket SDK, Azure Quantum	Apache 2.0	No
QCW Forge	SDK	Python	"Forge is a service that lets you develop and run quantum software. An interface to this service is exposed to Python developers via the Forge client library."	IBM Quantum, Rigetti, IonQ	Proprietary	No
ProjectQ	SDK	Python	"ProjectQ is an open source effort for quantum computing."	IBM Quantum, AQT, IonQ, Amazon Braket SDK, Azure Quantum	Apache 2.0	No
TensorFlow Quantum	SDK	Python	"TensorFlow Quantum (TFQ) is a quantum machine learning library for rapid prototyping of hybrid quantum-classical ML models."	Rigetti, AQT, IonQ, Quantinuum, Pasqal	Apache 2.0	No
PennyLane	SDK	Python	"[..] a cross-platform Python library for programming quantum computers. [..] connects quantum computing with powerful machine learning frameworks."	Qiskit, Amazon Bracket, Cirq, Microsoft QDK, Honeywell, IonQ, AQT, Rigetti Forest, Qualcs, Orquestra, Quantum Inspire, ProjectQ and Strawberry Fields	Apache 2.0	Yes

Our aim was to make the results publicly available for the researchers of HUN-REN Cloud, in a form enabling them to start experimenting with quantum resources quickly. We set up multiple requirements: (i) the published solution should include every component necessary to start developing quantum applications quickly; (ii) it should include a number of examples demonstrating the usability of quantum resources; and finally, (iii) it should be easily extensible with support for additional quantum resources.

A. SELECTED COMPONENTS

Based on the requirements defined, it is deemed natural that the target reference architecture should have a graphical gateway and development environment. In general, *science gateways* are web-based platforms that provide researchers with user-friendly access to computational resources, data, and scientific tools. In our case there are multiple candidates that fit this role: self-hosted solutions such as Visual Studio Code, JupyterLab or Apache Zeppelin or cloud-based ones such as Google Colab. We selected JupyterLab for the role of the gateway, as it is a de facto standard among scientists, its notebook functionality has an extensive feature set, is easy to deploy in varying environments, integrates very well with different tools (like Matplotlib), and it does not require external services.

Additionally, we wanted to include examples presenting the usability of quantum computers through solving some "real-life" problems. For this, we have selected the clustering problem demonstration of D-Wave. This problem can be solved with the help of Apache Spark as well, in its machine learning library through the K-mean algorithm. The aim of this demonstration is that we can show that such a problem can be solved with Apache Spark, but with quantum resources as well. In order to host this example, the reference architecture should include an Apache Spark cluster as well. For this we extended the existing Apache Spark reference architecture [33] provided by HUN-REN Cloud.

Besides serving as an alternate platform for the examples, Apache Spark also allows the exploitation of multiple nodes for computation. If the application execution could benefit from the usage of quantum resources, then with the help of quantum libraries deployed onto Spark worker nodes, this option also becomes available.

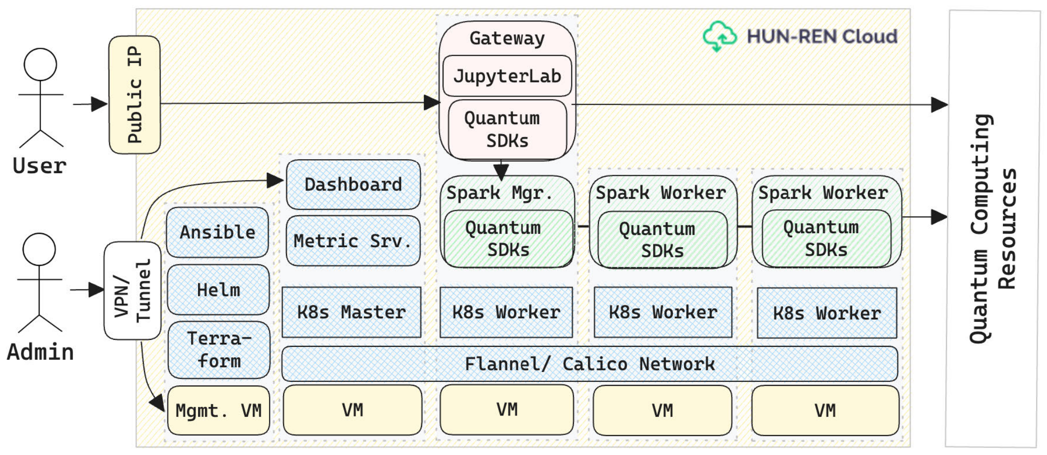


FIGURE 1. The Quantum - Science Gateway Hybrid Reference Architecture running on HUN-REN Cloud (components provided by the cloud are marked with yellow) utilizing the (i) Kubernetes Reference Architecture [64] (related components marked with blue grid); and extending the (ii) Apache Spark Reference Architecture [33] (related components marked with green stripes).

B. REFERENCE ARCHITECTURE IMPLEMENTATION

The reference architecture is software container based with the option to deploy the components on the hosts directly if needed. In order to speed up deployment, and provide portability, we recommend an orchestrated deployment of the reference architecture either via Docker Compose or Kubernetes. The following images are used to deploy the reference architecture: (1) the Science Gateway image containing JupyterLab, SDKs and examples; (2) the Apache Spark Master image; and (3) on or more instance of the image for the Apache Spark worker. The high-level component structure of the reference architecture is presented in Figure 2.

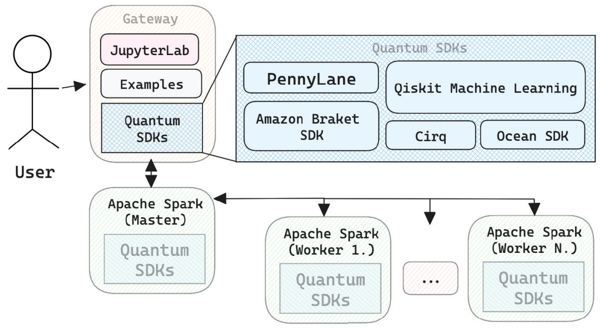


FIGURE 2. Component structure of the quantum reference architecture.

The *Science Gateway* component acts as the entry point for the users, thus, must include all the quantum computing SDKs pre-deployed, and also the ready-to-use examples. In order to build the image, we have selected Docker. With Docker, it is necessary to have a base image, which is used as a starting point for creating the component's final image. For JupyterLab, there are multiple base selection possibilities:

1) a small base image, for example a minimal Debian image. In this case it is necessary to deploy all the additional components (like JupyterLab, Spark libraries)
2) a specific base image, containing a set of prerequisites already deployed (like JupyterLab itself, or Java for Spark)

The base of the image selected was the OpenJDK Docker image (`openjdk:8-jdk-slim`), as Java is required to run the examples using Apache Spark. A Dockerfile describes the deployment steps of the image, these include the followings: installation of Python packages, installation of JupyterLab, installation of quantum SDKs, the deployment of example notebooks, and the deployment of the startup script. The task of the startup script is to create the default Jupyter configuration, set the password (based on an environment variable), and to start JupyterLab.

The component includes the following quantum SDKs: D-Wave's Ocean SDK, Amazon Braket SDK, Cirq and Qiskit SDK. Additionally, application-specific quantum libraries are also deployed: Qiskit Machine Learning, Qiskit Runtime, and PennyLane. These can be used to demonstrate the usability of quantum resources in machine learning applications.

Additionally, Spark Python packages are also deployed, allowing the execution of examples using the Apache Spark cluster. Two node types are used to build the *Apache Spark cluster*. a master node, and a worker node type, where there is one master node instance, but there can be multiple worker node instances. Similarly to the Gateway component, Docker was selected to build the container images. Also, the base Docker image is OpenJDK, as Java is required for Spark. Two Dockerfiles describe the master and the worker nodes. Both of these include the same instructions to deploy Apache Spark, however the way the containers start are different: the master runs the Spark Master class, while workers run the Spark Worker class. Beside Spark dependencies, all Spark nodes have the same set of quantum SDKs deployed as the JupyterLab component. As mentioned earlier, this allows the exploitation of quantum resources from applications run through Apache Spark.

C. COMPONENT COMPOSITION

At this point the container images representing the different components are ready, and in the next step they must be orchestrated. There are multiple options for orchestration, e.g., Docker (via Docker Compose or Swarm), Kubernetes, or virtual machines (via Terraform). Docker itself is a basic tool for bringing up individual containers, however requires additional work to bring up multiple containers which can communicate with each other. Docker Compose is an easy to use tool for bringing up composition of multiple containers, offering replication as well. It also allows the users to deploy the service stack onto a Docker Swarm cluster. Alternatively, Kubernetes is a widespread tool for managing large-scale service stacks containing multiple container instances, however it requires notable resources to deploy, and extensive operation knowledge. Finally, Terraform is a versatile orchestration tool supporting various cloud providers, and its services.

Based on the above, and the ease of the tools' usability, we decided to offer a Docker Compose and Kubernetes based deployment method for the users and Terraform can be used for provisioning virtual machines. In general, Docker Compose act as a "Swiss Army knife": users can easily start up the reference architecture in a Docker-capable environment, like a standalone Docker Engine deployment, a Docker Swarm cluster or on Kubernetes with Kompose [65].

Multiple deployment options are provided for the reference architecture: a base one, and a full one. The base deployment includes only the JupyterLab component, and this is the default one. In this case users do not have the possibility to run examples using Apache Spark, but can use all the other examples using quantum resources. On the other hand, the full deployment contains the Spark Master node, and a configurable number of Spark Worker nodes, allowing users to run all the provided examples.

D. AVAILABILITY OF THE REFERENCE ARCHITECTURE

The developed reference architecture is available through the HUN-REN Cloud portal [66], and has its own Git repository [67], hosted on a GitLab deployment. This repository includes all the necessary code to reproduce the components' images. Additionally, the CI/CD functionality of GitLab is used to build and publish new versions of the images when the code is updated. The Git repository also contains a short documentation on how to deploy the reference architecture in the HUN-REN Cloud. Furthermore, a Wiki page [68] is continuously maintained with documentation on how to get access to the different quantum hardware providers' resources.

IV. DEPLOYMENT AND USABILITY EXPERIENCES

In this section the deployment possibilities of the reference architecture in various environments are examined, and the deployment's usability is also evaluated.

The reference architecture supports both single node and orchestrated deployments. For single node deployment *Docker Compose* is recommended. However, the reference architecture can be directly deployment on any host or VM as well. Further, single-node Kubernetes deployment using e.g., Minikube, is also possible. For orchestrated or multi-node deployments Kubernetes is recommended tool. Either an existing Kubernetes cluster, including hosted services such as Azure Kubernetes Service can be used, or a new cluster can be deployed via the Kubernetes Reference Architecture [64] by HUN-REN Cloud (see Figure 1). The latter deployment is detailed on Figure 1. For legacy reasons Docker Swarm [69] based deployment is also supported.

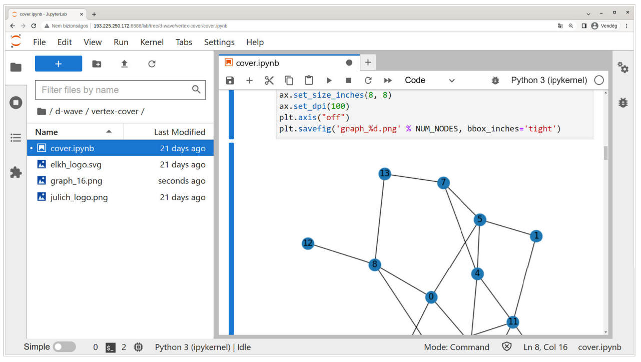
The deployment of the reference architecture was performed in the following environments: a user's local machine (for example, a laptop), the HUN-REN Cloud and Amazon Web Services. The deployment options are described in the following parts.

A. LOCAL DEPLOYMENT

In this scenario the user is using its own machine for deploying the reference architecture. The assumption in this scenario is that the user is using a recent Linux operating system, with Docker Engine and Docker Compose deployed. The steps to start the reference architecture in this case are as follows:

1) the user must clone the Git repository of the reference architecture; and
2) the user must start the requested version (either the base one using the default compose file, or the complete one using the full compose file).

This simple deployment method starts only the gateway component (JupyterLab), and the interface is accessible via HTTP on port 8888/TCP using the password defined in the `docker-compose.yml` file.



Name	Last Modified
cover.ipynb	21 days ago
elkh_logo.svg	21 days ago
graph_16.png	seconds ago
julich_logo.png	21 days ago

```
ax.set_size_inches(8, 8)
ax.set_dpi(100)
plt.axis("off")
plt.savefig('graph_%d.png' % NUM_NODES, bbox_inches='tight')
```

FIGURE 3. **The Gateway (JupyterLab) of the reference architecture running in HUN-REN Cloud.**

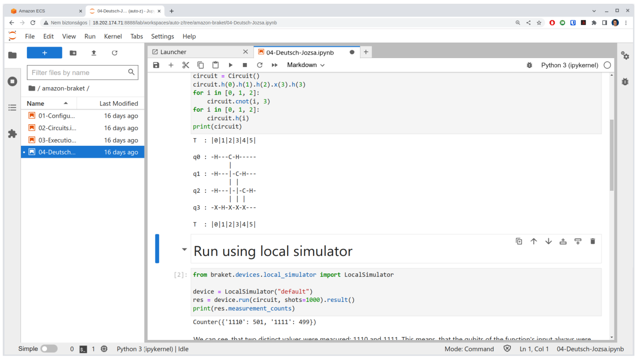
B. CLOUD-BASED DEPLOYMENTS

As the reference architecture aims at offering a quick-start package for experimenting with quantum resources for HUN-REN Cloud users, it is natural to check the deployment of the reference architecture in HUN-REN Cloud. This cloud offers Ubuntu 22.04 images for the users, which already has a Docker Engine and Docker Compose pre-deployed. On the top of this image, HUN-REN Cloud offers a Kubernetes reference [64] architecture (see Figure 1), which can be used to deploy the Quantum reference architecture in a distributed environment.

The extended Spark Reference Architecture uses the Standalone cluster manager type [70], regardless of the deployment mode (Docker single host, Docker multi-host or Kubernetes) to provide the same behavior on all deployments.

The user can follow the detailed documentation to deploy the Kubernetes cluster on HUN-REN Cloud. After the deployment they can easily deploy, the Quantum reference architecture on the top of the Kubernetes cluster with the help of the provided `docker-compose.yml` file and the Kompose tool or with the Kubernetes deployment files. After the deployment the Kuberetes master node will have a Floating IP, and the user can access the JupyterLab interface through that using the 8888 port. The examples are usable from this deployment as well, as shown in Figure 3.

Finally, we have validated the deployment of the reference architecture in Amazon Web Services. There are multiple deployment possibilities, such as creating a task in Amazon Elastic Container Service (ECS), using Docker Compose with a properly configured ECS context, using a hosted Kubernetes solution or just deploying in virtual machines. For testing, we selected the first method, as through the AWS console a container can be started up within a few minutes. Care must be taken when configuring networking: for the test using a public IP address and a security group configured to allow incoming traffic to TCP port 8888 was set up. Figure 4 shows the gateway accessible via the public IP address running the example notebook demonstrating the *Deutsch-Jozsa algorithm* [71].


Amazon ECS
04-Deutsch-J... (auto-z) - Jup
Nem biztonságos | 18.202.174.71:8888/lab/workspaces/auto-z/tree/amazon-braket/04-Deutsch-Jozsa.ipynb
File
Edit
View
Run
Kernel
Tabs
Settings
Help
Filter files by name
/ amazon-braket /
Name
Last Modified
01-Configu... 16 days ago
02-Circuits.i... 16 days ago
03-Executio... 16 days ago
04-Deutsch... 16 days ago
Launcher
04-Deutsch-Jozsa.ipynb
Markdown
Python 3 (ipykernel)
Simple
Python 3 (ipykernel) | Idle
Mode: Command
Ln 1, Col 1
04-Deutsch-Jozsa.ipynb

```
circuit = Circuit()
circuit.h(0).h(1).h(2).x(3).h(3)
for i in [0, 1, 2]:
    circuit.cnot(i, 3)
for i in [0, 1, 2]:
    circuit.h(i)
print(circuit)
```

```
T  : |0|1|2|3|4|5|

q0 : -H---C-H-----
          |
q1 : -H---|-C-H---
          | |
q2 : -H---|-|-C-H-
          | | |
q3 : -X-H-X-X-X---

T  : |0|1|2|3|4|5|
```

Run using local simulator

```
[2]: from braket.devices.local_simulator import LocalSimulator

     device = LocalSimulator("default")
     res = device.run(circuit, shots=1000).result()
     print(res.measurement_counts)
```

```
Counter({'1110': 501, '1111': 499})
```

We can see, that two distinct values were measured: 1110 and 1111. This means, that the qubits of the function's input always were

FIGURE 4. The gateway (JupyterLab) of the reference architecture running in Amazon Web services.

C. USABILITY EXPERIENCES

Finally, the usability of a deployment of the reference architecture was tested. In this scenario the connection to the following quantum resources and usability of the mentioned frameworks was tested: IBM Quantum with Qiskit, Amazon Braket (using IonQ) and D-Wave (with the Vertex Cover problem example).

TABLE 4. Measurements of the Deutsch-Jozsa circuit using the IonQ [4] QPU through Amazon braket.

Case	1110	1111	1100	1010	1000	0110
Measurement Count	48	46	2	2	1	1

For testing IBM Quantum access, a prepared Qiskit example in the reference architecture was used, the `03-Execution` notebook. This example prepares a quantum circuit with 4 qubits, applies Hadamard gates to each of them, and finally takes measurements. As all of the qubits are in the equal superposition state, the measurements should return random numbers in the range of 0, . . . , 15. The example runs the circuit using the local `aer_simulator`, the cloud-based `simulator_stabilizer`, and finally, a real QPU device, `ibm_oslo`. In every case, the circuit was run using multiple shots. We could see that the example was properly executed, however the queuing time for the executions using QPU device were around 10 minutes.

Testing Amazon Braket also did rely on one of the examples, namely the `04-Deutsch-Jozsa` notebook. This example demonstrates the Deutsch-Jozsa algorithm's [71] usage with a balanced function, operating on 3 qubits. The circuit has a total of 4 qubits, the fourth one being the result of the function. According the Deutsch-Jozsa algorithm, for a balanced function we should measure all input parameter bits as 1. When using the local simulator in the example, two values were measured: `1110` and `1111`, which matches our expectations: the first three bits are always 1. However, when using a real QPU device, the IonQ provided by Amazon Braket, based on 100 shots additional values were also measured. Table 4 shows the number of occurrences for the different measurements. In contrary to the ideal case, we can see some invalid results as well, this is the consequence of the erroneous property of current QPU devices, which should be handled following some error mitigation methods [72], [73].

The final test scenario covers the usability and some benchmarking tests of the D-Wave resources, through the minimum vertex cover problem example, examining the processing time required to determine the minimum vertex cover of Erdős-Rényi graphs with increasing number of nodes. The example uses the local (CPU-based) `ExactSolver` and the QPU-backed `DWaveSampler` to get solutions for the problem. As the last step, the example presents a diagram of the processing times required using the different devices. An example result is shown for 10..35 nodes in Figure 5. As it can be seen, when local resource is used, the processing time grows exponentially as the node number increases. While using the QPU devices requires basically constant time.

V. FUTURE WORK

The development of the reference architecture is an important step in the process of creating a quantum-science gateway to make quantum computing much easier and more accessible for researchers who want to use this very advanced tool in their own research. Considering that both quantum

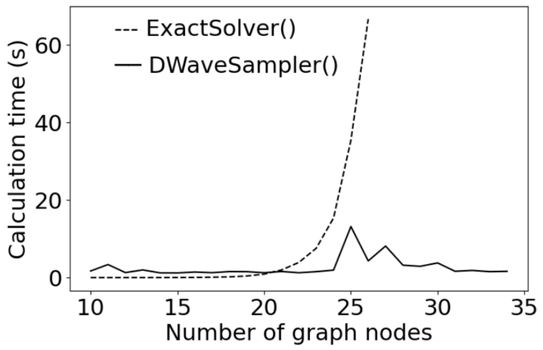


FIGURE 5. CPU and QPU processing times for the vertex cover problem.

hardware and quantum algorithms and applications are novel, rapidly evolving, experimental fields and that in many cases different implementations support the tasks with different efficiencies, it is important that prospective users can easily test and compare the possibilities. The reference architecture described above enables this by providing a free, single, open and portable environment from which many different quantum hardware and simulators can be accessed in a similar way.

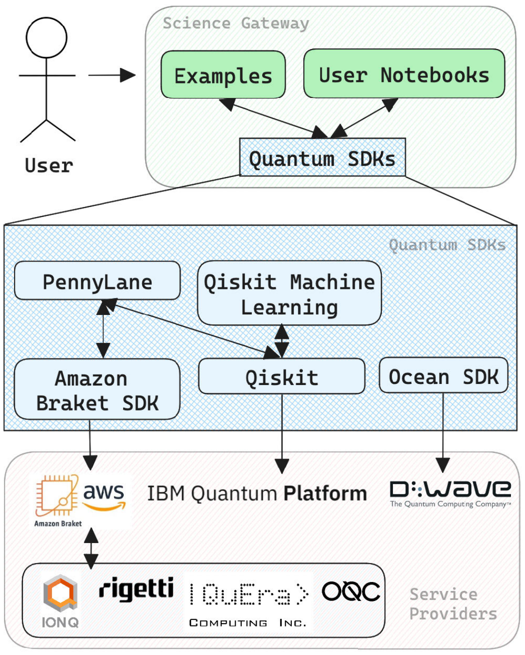


FIGURE 6. QPU access possibilities in the hybrid reference architecture.

The reference architecture is flexible enough to be deployed in various environments: in local environments, in community clouds such as HUN-REN Cloud, or at public cloud providers like Amazon Web Services. A modular software container-based approach and the utilization of standardized orchestration methods make this possible.

Additionally, multi-user support is also considered in the form of using JupyterHub as the gateway. Currently, the JupyterLab component allows a single-user mode, thus multiple deployments of the reference architecture are required to serve multiple users. However, with JupyterHub, multiple users could exploit the possibilities of the quantum reference architecture by using a single deployment. Having a multi-user gateway is not enough, as the backend services (e.g., Apache Spark) must be multi-user aware, for example a batch scheduler such as Hadoop Yarn or the native Kubernetes scheduler must be configured for them.

Finally, as the application possibilities of quantum resources increases through the improvement of the different frameworks, we are also continuously maintaining the SDKs and frameworks integrated in the reference architecture.

VI. CONCLUSION

In this paper we presented a novel approach for providing access to a diverse set of real quantum resources. Our hybrid reference architecture incorporates major quantum software development kits (SDKs) and frameworks, such as those provided by IBM, D-Wave, IonQ, and quantum devices available through Amazon Braket (e.g., IonQ or Rigetti QPU devices). The architecture also contain frameworks enabling experiments with quantum resources in the field of machine learning: Qiskit Machine Learning and PennyLane. The content of the architecture is visually summarized in Figures 1, 2 and 6. We also provide a set of examples and present preliminary benchmark results, these can serve as practical guides for constructing solutions to other (predefined) problems. On top of these, our reference architecture aims to provide an user-friendly interface for researchers.

We hope these combined will democratize access to quantum computing resources, and thus promote scientific collaboration.

ACKNOWLEDGMENT

The authors would like to express their sincere gratitude to Zoltán Farkas and Miklós Kozlovszky for their invaluable comments and feedback shaping this study. On behalf of the "Quantum Reference Architecture" project we are grateful for the possibility to use HUN-REN Cloud (see [1]; https://science-cloud.hu/) which helped us achieve the results published in this paper.

REFERENCES

[1] M. Héder, E. Rigó, D. Medgyesi, R. Lovas, S. Tenczer, F. Török, A. Farkas, M. Emodi, J. Kadlecsik, Á. Pintér, and P. Kacsuk, "The past, present and future of the ELKH cloud," *Információs Társadalom*, vol. 22, no. 2, p. 128, Aug. 2022.

[2] P. I. Bunyk, E. M. Hoskinson, M. W. Johnson, E. Tolkacheva, F. Altomare, A. J. Berkley, R. Harris, J. P. Hilton, T. Lanting, A. J. Przybysz, and J. Whittaker, "Architectural considerations in the design of a superconducting quantum annealing processor," *IEEE Trans. Appl. Supercond.*, vol. 24, no. 4, pp. 1–10, Aug. 2014.

[3] K. Boothby, C. Enderud, T. Lanting, R. Molavi, N. Tsai, M. H. Volkmann, F. Altomare, M. H. Amin, M. Babcock, A. J. Berkley, and C. B. Aznar, "Architectural considerations in the design of a third-generation superconducting quantum annealing processor," 2021, *arXiv:2108.02322*.

[4] S. Debnath, N. M. Linke, C. Figgatt, K. A. Landsman, K. Wright, and C. Monroe, "Demonstration of a small programmable quantum computer with atomic qubits," *Nature*, vol. 536, no. 7614, pp. 63–66, Aug. 2016.

[5] J. M. Amini, Y. Nam, N. Grzesiak, J.-S. Chen, N. C. Pisenti, M. Chmielewski, and C. Collins, "Benchmarking an 11-qubit quantum computer," *Nature Commun.*, vol. 10, no. 1, p. 5464, 2019.

[6] Y. Alexeev, D. Bacon, K. R. Brown, R. Calderbank, L. D. Carr, F. T. Chong, B. DeMarco, D. Englund, E. Farhi, B. Fefferman, and A. V. Gorshkov, "Quantum computer systems for scientific discovery," *PRX Quantum*, vol. 2, no. 1, Feb. 2021, Art. no. 017001.

[7] V. Hassija, V. Chamola, V. Saxena, V. Chanana, P. Parashari, S. Mumtaz, and M. Guizani, "Present landscape of quantum computing," *IET Quantum Commun.*, vol. 1, no. 2, pp. 42–48, Dec. 2020.

[8] N. Killoran, J. Izaac, N. Quesada, V. Bergholm, M. Amy, and C. Weedbrook, "Strawberry fields: A software platform for photonic quantum computing," *Quantum*, vol. 3, p. 129, Mar. 2019.

[9] S. J. Devitt, "Performing quantum computing experiments in the cloud," *Phys. Rev. A, Gen. Phys.*, vol. 94, no. 3, Sep. 2016, Art. no. 032329.

[10] H. T. Nguyen, M. Usman, and R. Buyya, "IQuantum: A case for modeling and simulation of quantum computing environments," 2023, *arXiv:2303.15729*.

[11] F. Leymann, J. Barzen, M. Falkenthal, D. Vietz, B. Weder, and K. Wild, "Quantum in the cloud: Application potentials and research opportunities," 2020, *arXiv:2003.06256*.

[12] P. Krantz, M. Kjaergaard, F. Yan, T. P. Orlando, S. Gustavsson, and W. D. Oliver, "A quantum engineer's guide to superconducting qubits," *Appl. Phys. Rev.*, vol. 6, no. 2, Jun. 2019, Art. no. 021318.

[13] *What is a Reference Architecture—Enterprise it Definitions*. Accessed: Jul. 23, 2023. [Online]. Available: https://www.hpe.com/us/en/what-is/reference-architecture.html

[14] P. Pääkkönen and D. Pakkala, "Reference architecture and classification of technologies, products and services for big data systems," *Big Data Res.*, vol. 2, no. 4, pp. 166–186, Dec. 2015.

[15] *Microsoft Azure Documentation—Reference Architectures*. Accessed: Jul. 23, 2023. [Online]. Available: https://docs.microsoft.com/en-us/azure/architecture/browse/

[16] *The TOGAF Standard, Version 9.2 Overview*. Accessed: Feb. 12, 2022. [Online]. Available: https://www.opengroup.org/togaf

[17] R. Cloutier, G. Müller, D. Verma, R. Nilchiani, E. Hole, and M. Bone, "The concept of reference architectures," *Syst. Eng.*, vol. 13, no. 1, pp. 14–27, Mar. 2010.

[18] M. Weyrich and C. Ebert, "Reference architectures for the Internet of Things," *IEEE Softw.*, vol. 33, no. 1, pp. 112–116, Jan. 2016.

[19] A. C. Marosi, M. Emodi, Á. Hajnal, R. Lovas, T. Kiss, V. Poser, J. Antony, S. Bergweiler, H. Hamzeh, J. Deslauriers, and J. Kovács, "Interoperable data analytics reference architectures empowering digital-twin-aided manufacturing," *Future Internet*, vol. 14, no. 4, p. 114, Apr. 2022.

[20] P. Mell and T. Grance, "The NIST definition of cloud computing," Special Publication (NIST SP), Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Sep. 2011. [Online]. Available: https://doi.org/10.6028/NIST.SP.800-145

[21] O. Sefraoui, M. Aissaoui, and M. Eleuldj, "OpenStack: Toward an open-source solution for cloud computing," *Int. J. Comput. Appl.*, vol. 55, no. 3, pp. 38–42, Oct. 2012.

[22] D. Milojicic, I. M. Llorente, and R. S. Montero, "OpenNebula: A cloud management tool," *IEEE Internet Comput.*, vol. 15, no. 2, pp. 11–14, Mar. 2011.

[23] S. Mathew and J. Varia, "Overview of Amazon web services," *Amazon Whitepapers*, vol. 105, pp. 1–22, Jan. 2014.

[24] R. Jennings, *Cloud Computing With the Windows Azure Platform*. Hoboken, NJ, USA: Wiley, 2010.

[25] M. Copeland, J. Soh, A. Puca, M. Manning, D. Gollob, M. Copeland, J. Soh, A. Puca, M. Manning, and D. Gollob, "Microsoft Azure and cloud computing," in *Microsoft Azure*. Berkeley, CA, USA: Apress, 2015, pp. 3–26.

[26] E. Bisong, "An overview of Google cloud platform services," in *Building Machine Learning and Deep Learning Models on Google Cloud Platform*. Berkeley, CA, USA: Apress, 2019, pp. 7–10.

[27] A. Kochut, Y. Deng, M. R. Head, J. Munson, A. Sailer, H. Shaikh, C. Tang, A. Amies, M. Beaton, D. Geiss, and D. Herman, "Evolution of the IBM cloud: Enabling an enterprise cloud services ecosystem," *IBM J. Res. Develop.*, vol. 55, no. 6, pp. 1–13, Nov. 2011.

[28] *Microsoft Azure Documentation—Reference Architectures for Data Analytics*. Accessed: Aug. 8, 2023. [Online]. Available: https://learn.microsoft.com/en-us/azure/architecture/browse/?azure%20categories=analytics&azure_categories=analytics

[29] *Aws Architecture Center—Architecture Best Practices for Analytics & Big Data*. Accessed: Aug. 8, 2023. [Online]. Available: https://aws.amazon.com/architecture/analytics-big-data/?cards-all.sort-by=item.additionalFields.sortDate&cards-all.sort-order=desc&awsf.content-type=content-type%23reference-arch-diagram&awsf.methodology=*all

[30] S.-W. Lin, B. Murphy, E. Clauer, U. Loewen, R. Neubert, G. Bachmann, M. Pai, and M. Hankel, "Architecture alignment and interoperability: An industrial internet consortium and platform industrie 4.0 joint whitepaper," White Paper, Industrial Internet Consortium, Boston, MA, USA, Tech. Rep. IIC:WHT:IN3:V1.0:PB:20171205, 2017. [Online]. Available: https://www.iiconsortium.org/pdf/JTG2_Whitepaper_final_20171205.pdf

[31] K. Schweichhart. (2016). *Reference Architectural Model Industrie 4.0 (RAMI 4.0)*. [Online]. Available: https://www.plattform-i40.deI

[32] A. Farkas, K. Póra, S. Szénási, G. Kertész, and R. Lovas, "Evaluation of a distributed deep learning framework as a reference architecture for a cloud environment," in *Proc. IEEE 10th Jubilee Int. Conf. Comput. Cybern. Cyber-Medical Syst. (ICCC)*, Jul. 2022, pp. 83–88.

[33] E. Nagy, R. Lovas, I. Pintye, Á. Hajnal, and P. Kacsuk, "Cloud-agnostic architectures for machine learning based on apache spark," *Adv. Eng. Softw.*, vol. 159, Sep. 2021, Art. no. 103029.

[34] *Terraform by Hashicorp: Terraform is an Open-Source Infrastructure as Code Software Tool That Enables You to Safely and Predictably Create, Change, and Improve Infrastructure*. Accessed: Mar. 22, 2022. [Online]. Available: https://www.terraform.io

[35] *Ansible is Simple it Automation*. Accessed: Mar. 22, 2022. [Online]. Available: https://www.ansible.com

[36] S. S. Gill, A. Kumar, H. Singh, M. Singh, K. Kaur, M. Usman, and R. Buyya, "Quantum computing: A taxonomy, systematic review and future directions," *Softw., Pract. Exper.*, vol. 52, no. 1, pp. 66–114, Jan. 2022.

[37] S. Morita and H. Nishimori, "Mathematical foundation of quantum annealing," *J. Math. Phys.*, vol. 49, no. 12, Dec. 2008, Art. no. 125210.

[38] P. Hauke, H. G. Katzgraber, W. Lechner, H. Nishimori, and W. D. Oliver, "Perspectives of quantum annealing: Methods and implementations," *Rep. Prog. Phys.*, vol. 83, no. 5, May 2020, Art. no. 054401.

[39] A. C.-C. Yao, "Quantum circuit complexity," in *Proc. IEEE 34th Annu. Found. Comput. Sci.*, Feb. 1993, pp. 352–361.

[40] M. Benedetti, E. Lloyd, S. Sack, and M. Fiorentini, "Parameterized quantum circuits as machine learning models," *Quantum Sci. Technol.*, vol. 4, no. 4, Nov. 2019, Art. no. 043001.

[41] D. P. DiVincenzo, "Quantum gates and circuits," *Proc. Roy. Soc. London A, Math., Phys. Eng. Sci.*, vol. 454, no. 1969, pp. 261–276, 1998.

[42] B. Heim, M. Soeken, S. Marshall, C. Granade, M. Roetteler, A. Geller, M. Troyer, and K. Svore, "Quantum programming languages," *Nature Rev. Phys.*, vol. 2, no. 12, pp. 709–722, 2020.

[43] D. Willsch, M. Willsch, C. D. Gonzalez Calaza, F. Jin, H. De Raedt, M. Svensson, and K. Michielsen, "Benchmarking advantage and D-wave 2000Q quantum annealers with exact cover problems," *Quantum Inf. Process.*, vol. 21, no. 4, p. 141, Apr. 2022.

[44] *D-Wave Ocean SDK*. Accessed: Oct. 24, 2023. [Online]. Available: https://docs.ocean.dwavesys.com/

[45] *Google Quantum AI*. Accessed: Oct. 3, 2023. [Online]. Available: https://quantumai.google

[46] *IBM Quantum Computing*. Accessed: Oct. 3, 2023. [Online]. Available: https://www.ibm.com/quantum, Accessed 2023-10-03.

[47] *IQM Superconducting Quantum Computer*. Accessed Oct. 24, 2023. [Online]. Available: https://meetiqm.com/

[48] OxfordQuantumCircuits. Accessed: Oct. 3, 2023. [Online]. Available: https://oxfordquantumcircuits.com

[49] W. Zeng, B. Johnson, R. Smith, N. Rubin, M. Reagor, C. Ryan, and C. Rigetti, "First quantum computers need smart software," *Nature*, vol. 549, no. 7671, pp. 149–151, Sep. 2017.

[50] J. Olivares-Sánchez, J. Casanova, E. Solano, and L. Lamata, "Measurement-based adaptation protocol with quantum reinforcement learning in a Rigetti quantum computer," *Quantum Rep.*, vol. 2, no. 2, pp. 293–304, May 2020.

[51] *AQT Ion Trap Quantum Computer*. Accessed: Oct. 24, 2023. [Online]. Available: https://www.aqt.eu/

[52] J. M. Pino, J. M. Dreiling, C. Figgatt, J. P. Gaebler, S. A. Moses, M. S. Allman, C. H. Baldwin, M. Foss-Feig, D. Hayes, K. Mayer, C. Ryan-Anderson, and B. Neyenhuis, "Demonstration of the trapped-ion quantum CCD computer architecture," *Nature*, vol. 592, no. 7853, pp. 209–213, Apr. 2021.

[53] *PASQAL Neutral Atoms Quantum Computers*. Accessed: Oct. 24, 2023. [Online]. Available: https://www.pasqal.com//
[54] *QuEra Neutral Atoms Quantum Computers*. Accessed: Oct. 24, 2023. [Online]. Available: https://www.quera.com///
[55] J. Wurtz, A. Bylinskii, B. Braverman, J. Amato-Grill, S. H. Cantu, F. Huber, A. Lukin, F. Liu, P. Weinberg, J. Long, and S.-T. Wang, "Aquila: Quera's 256-qubit neutral-atom quantum computer," 2023, *arXiv:2306.11727*.
[56] *Quantum Cloud Service—Quantum Computing Service—Amazon Braket—AWS*. Accessed: Oct. 5, 2023. [Online]. Available: https://aws.amazon.com/braket/
[57] *Aws: Create an Amazon Braket Notebook Instance*. Accessed: Nov. 1, 2023. [Online]. Available: https://docs.aws.amazon.com/braket/latest/developerguide/braket-get-started-create-notebook.html
[58] *Google Quantum AI: Cirq*. Accessed: Oct. 31, 2023. [Online]. Available: https://quantumai.google/cirq
[59] M. Steffen, D. P. DiVincenzo, J. M. Chow, T. N. Theis, and M. B. Ketchen, "Quantum computing: An IBM perspective," *IBM J. Res. Develop.*, vol. 55, no. 5, pp. 1–11, Sep. 2011.
[60] D. García-Martín and G. Sierra, "Five experimental tests on the 5-Qubit IBM quantum computer," *J. Appl. Math. Phys.*, vol. 6, no. 7, pp. 1460–1475, 2018.
[61] A. Cross, "The IBM Q experience and QISKit open-source quantum computing software," in *Proc. APS March Meeting Abstr.*, 2018, p. 58.
[62] *IBM Quantum Composer*. Accessed: Oct. 3, 2023. [Online]. Available: https://quantum-computing.ibm.com/composer/
[63] *IBM Quantum Learning Lab*. Accessed: Oct. 3, 2023. [Online]. Available: https://lab.quantum-computing.ibm.com/
[64] *Kuberentes Reference Architecture*. Accessed: Oct. 3, 2023. [Online]. Available: https://science-cloud.hu/en/reference-architectures/kubernetes-cluster
[65] *Kompose*. Accessed: Oct. 3, 2023. [Online]. Available: https://kompose.io/
[66] *Quantum Reference Architecture*. Accessed: Feb. 27, 2023. [Online]. Available: https://science-cloud.hu/en/reference-architectures/quantum
[67] *Quantum Reference Architecture, Git repository*. Accessed: Feb. 27, 2023. [Online]. Available: https://git.sztaki.hu/science-cloud/reference-architectures/quantum
[68] *Quantum Reference Architecture, Wiki Page*. Accessed: Feb. 27, 2023. [Online]. Available: https://git.sztaki.hu/science-cloud/reference-architectures/quantum/-/wikis/home
[69] *Docker: Swarm Mode Overview*. Accessed: Oct. 31, 2023. [Online]. Available: https://docs.docker.com/engine/swarm/
[70] *Apache Spark: Cluster Mode Overview*. Accessed: Oct. 30, 2022. [Online]. Available: https://spark.apache.org/docs/latest/cluster-overview.html
[71] D. Deutsch and R. Jozsa, "Rapid solution of problems by quantum computation," *Proc. Roy. Soc. London A, Math. Phys. Sci.*, vol. 439, no. 1907, pp. 553–558, Dec. 1992.
[72] M. Scheerer, J. Klamroth, and O. Denninger, "Fault-tolerant hybrid quantum software systems," in *Proc. IEEE Int. Conf. Quantum Softw. (QSW)*, Jul. 2022, pp. 52–57.
[73] J. D. Guimarães and C. Tavares, "Towards a layered architecture for error mitigation in quantum computation," in *Proc. IEEE Int. Conf. Quantum Softw. (QSW)*, Jul. 2022, pp. 41–51.

ATTILA CSABA MAROSI received the M.Sc. and Ph.D. degrees from the Budapest University of Technology and Economics. He is currently a Research Fellow with the Laboratory of Parallel and Distributed Systems (LPDS), Institute for Computer Science and Control (SZTAKI), Hungarian Research Network (HUN-REN). He has more than 20 years of research and development experience, involving both industry and academia that encompasses a wide range of distributed and parallel systems. His research interests include large-scale time-series data collection and inference. His latest research achievements in the fields of big data and cloud computing contribute to the research on Cooperative Production and Logistics Systems to Support a Competitive and Sustainable Economy (COPROLOGS) Project and the National Laboratory for Autonomous Systems (NLAS).

ATTILA FARKAS received the B.Sc. and M.Sc. degrees in computer science engineering from the John von Neumann Faculty of Informatics, Óbuda University. He is currently pursuing the Ph.D. degree in distributed deep learning. He is a Research Associate with the Laboratory of Parallel and Distributed Systems (LPDS), Institute for Computer Science and Control (SZTAKI), Hungarian Research Network (HUN-REN). He has been involved in COLA and NEANIAS European H2020 projects. His research interests include parallel computing, clouds, container technologies, and machine learning.

TAMÁS MÁRAY received the M.Sc. and Ph.D. degrees from the Budapest University of Technology and Economics. He is currently a Research Fellow with the Laboratory of Parallel and Distributed Systems (LPDS), Institute for Computer Science and Control (SZTAKI), Hungarian Research Network (HUN-REN). For more than ten years, he was a Professor with the Budapest University of Technology and Economics, and for 16 years he was the Technical Director of the NIIF Institute, which developed and operated the Hungarian Research and Education Network (NREN). He set up Hungary's first web server, in 1993 and played a key role in the development and spread of the Internet and its applications in Hungary. Under professional guidance, the first TOP500 grade supercomputer was implemented in Hungary, in 2001. He led the development and operation of the Hungarian national HPC infrastructure for two decades. The Hungarian HPC Competence Centre (HPC@hu) was established under leadership. He is a delegate in international HPC professional organizations. His research interests include software architectures for parallel systems, supercomputers, and quantum computers, networking technologies and protocols, and software design methodologies.

RÓBERT LOVAS received the Ph.D. degree in informatics from the Budapest University of Technology and Economics. He is currently the Deputy Director of the Institute for Computer Science and Control (SZTAKI), Hungarian Research Network (HUN-REN). He is also a Habilitated Associate Professor and the Founder of the Institute for Cyber-Physical Systems with the John von Neumann Faculty of Informatics, Óbuda University. His research and development experience in a wide range of application fields of distributed and parallel systems has been gained in various global, EU, and national collaborations with academic organizations, universities, and enterprises, focusing on computational chemistry, numerical meteorological modeling, bioinformatics, agriculture, connected cars, and Industry 4.0. He has been coordinating EU FP7/H2020/Horizon Europe projects and the HUN-REN Cloud Research Infrastructure. His latest cloud, big data, the IoT, and AI-related research achievements contribute to the recently launched Artificial Intelligence National Laboratory and the National Laboratory for Autonomous Systems. He is a member of the Committee on Information Science, Hungarian Academy of Sciences. Recently, he has been elected as an Executive Board Member of EGI.