IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 1, December 2025
Impact Factor: 7.67

Modeling and Performance Evaluation of Hybrid Classical–Quantum Serverless Computing Platforms

Wakade Kartik[1], Vyawahare Shubham[2], Mhase Dhananjay[3], Prof. P. V. Gaikwad[4]
Student, Department of Computer Engineering[123]
Professor, Dept, of Computer Engineering[4]
Adsul's Technical Campus, Chas, Ahilyanagar, Maharashtra, India

Abstract: *While quantum computing technologies are evolving toward achieving full maturity, hybrid algorithms, such as variational quantum computing, are already emerging as valid candidates to solve practical problems in fields, such as chemistry and operations research. This situation calls for a tighter and better integration of classical and quantum computing infrastructures to improve efficiency and users' quality of service. Inspired by recent developments in cloud technologies, serverless computing has recently been considered a promising solution for this purpose by both industry and research. In this work, we define a system model for a hybrid classical–quantum serverless system, with an associated open-source numerical simulator that can be driven by production traces and stochastic workload models. We therefore describe how we produced a public dataset using IBM Qiskit in a local and remote infrastructure, with a sample application on optimization. The simulation results show initial insights on some distinguishing features of the platform simulated, measured in terms of user and system metrics, for jobs with heterogeneous problem sizes and priorities. We also report a few lessons we learned from developing the application with IBM Qiskit serverless and running it on IBM Quantum backends*

Keywords*:* High-performance computing (HPC), hybrid computing, quantum approximate optimization algorithm (QAOA), quantum computing, quantum optimization, serverless computing, variational quantum computing, variational quantum eigensolver (VQE)

I. INTRODUCTION

Quantum computing is an emerging technology with the potential to revolutionize scientific research [1] and our industry, economy, and whole society [2]. However, despite recent significant investments and the growing trend of interest in the scientific community [3], the technology is not yet fully developed, with our current period being famously called noisy intermediate-scale quantum (NISQ) era. A class of applications that is particularly promising in the near term is variational quantum computing [4], which is an iterative method where "variational" parameters of a quantum application are adapted through an optimization process running in a classical computer. This technique can be used in many fields, including quantum machine learning (QML) [5], operations research [6], and chemistry [7].

The simplified architecture of a typical platform for running variational quantum computing jobs is illustrated in Fig. 1, showing the classical versus quantum task queues and computing resources and highlighting the hybrid classical–quantum nature of such a computing infrastructure. However, as the ecosystem of the quantum computing platform providers becomes richer and the maturity of the solutions they offer improves, we see a shift in the way platforms are designed, operated, and made available to the users [8]: software engineering is entering the field, with a promise to make systems more scalable, efficient, and easy to use [9]. One of the improvements proposed is adopting a serverless computing approach.

Serverless computing [10] is a mature technology in cloud services that enables developers to write applications as collections of elementary stateless functions calling one another. The functions run inside lightweight virtualization

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 1, December 2025
Impact Factor: 7.67

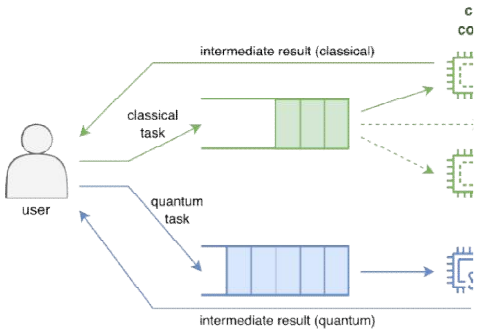


FIGURE 1. High-level representation of a traditional platform for executing hybrid classical–quantum applications. abstractions, usually containers, and are automatically scaled up when the demand for a given function increases, thereby spawning more workers that run the same function to which a load balancer dispatches invocations. On the other hand, when there are fewer function invocations, the platform progressively reduces the number of workers, down to zero, if necessary. System providers like serverless computing because of its inherent flexibility, which enables them to finetune the use of resources efficiently [11]. Users enjoy the programming model, called function as a service (FaaS), which relieves them from all management tasks and enables pay-per-use billing schemes [12].

A high-level serverless architecture for hybrid classical– quantum computing is illustrated in Fig. 2. The computing resources underneath are the same as those in Fig. 1, i.e., classical high performance computing (HPC) resources and QCs, but the application now consists of functions run by workers running in containers interacting with one another: the input of a function is the output of its predecessor, starting with a first invocation that was done by the user through an online gateway. The application logic carried out by the workers is also provided by the user as a container image interfaces (APIs) for the execution of serverless jobs in their quantum cloud services. However, there is a gap in the modeling and optimization of such an emerging approach,

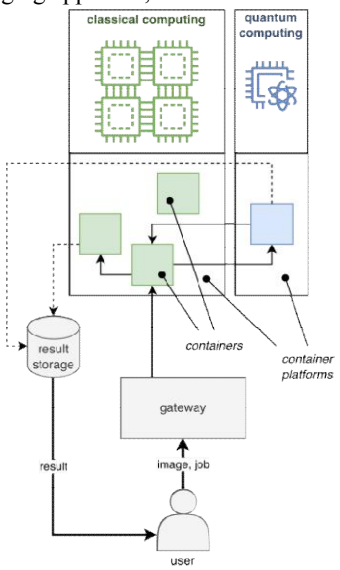


FIGURE 2. High-level representation of a hybrid classical–quantum serverless computing infrastructure

DOI: 10.48175/IJARSCT-30155

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 1, December 2025
Impact Factor: 7.67

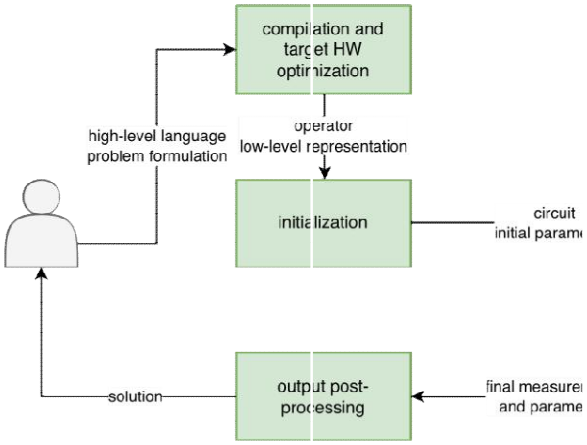


FIGURE 3. General flow diagram of a variational quantum computing algorithm, where the different tasks execute on quantum (cyan) versus classical

which we fill in this work. The rest of this article is organized as follows. In Section II, we propose a system model of a hybrid classical– quantum serverless platform for variational quantum computing. In Section III, we report results obtained from benchmarking experiments on a local Qiskit serverless deployment and IBM Quantum, which are used to drive the numerical simulations illustrated in Section IV, aimed at gathering an initial understanding of the impact of job priorities and different policies to schedule quantum tasks. The limitations of the current work and our lessons learned are discussed in Section V. Section VI surveys the relevant state of the art in the scientific literature. Finally, Section VII concludes this article.

II. SYSTEM MODEL

In this section, we illustrate the system model adopted. We proceed stepwise, by first introducing our target application,

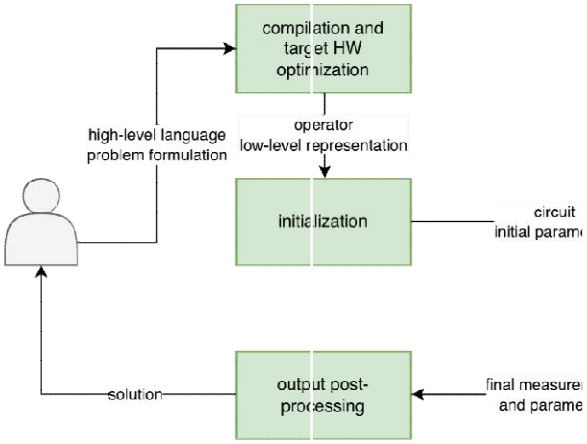


FIGURE 3. General flow diagram of a variational quantum computing algorithm, where the different tasks execute on quantum (cyan) versus classical (green) computing resources.

i.e., iterative quantum optimization, in Section II-A. Then, we provide a mathematical formulation of a traditional multiserver problem in Section II-B to set the scene with commonly used notation, before explaining how to go from there to the final serverless-oriented system model in Section II-C.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 1, December 2025
Impact Factor: 7.67

A. VARIATIONAL QUANTUM COMPUTING

As anticipated in Section I, in this work, we focus on a class of hybrid classical–quantum applications under the umbrella of variational quantum computing [13]. A high-level schematic of such applications is illustrated in Fig. 3. The flow begins with the user, represented in the leftmost part of the picture, formulating their problem using a highlevel language, typically Python. Such formulation, for instance, can capture an objective function to be optimized by complying with some constraints (quantum approximate optimization algorithm (QAOA) [6]) or model a physical system, such as a molecule, for which the ground state needs to be found (variational quantum eigensolver (VQE) [7]). The high-level formulation needs to be converted into a suitable low-level representation that depends on the target hardware. In the following, we assume for simplicity of explanation the latter to be a gate-based QC [14], in which case the low-level representation would be a quantum circuit that depends on a set of parameters, typically angles for rotating qubits along one of the axes.1 The resulting quantum circuit is then sent to a QC, along with an initial configuration of the parameters based on external/expert knowledge (or random data). After a single execution of the quantum circuit, the qubits are measured. This operation can be seen as the ground state of a certain Hamiltonian, as resulting in the mapping from the high-level problem formulation to the circuit representation. The measurements allow a classical task to compute the energy/cost

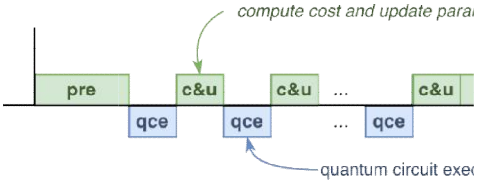


FIGURE 4. Time sequence of a variational quantum algorithm job.

of the system and, by using classical techniques only, to provide updated parameters for the quantum circuit to be reexecuted. These two steps are repeated until convergence (or, in practice, a maximum number of iterations). At this point, the final measurements and parameters are produced as output: through postprocessing, they will provide the user with a solution found.

The simplified time sequence of a single variational quantum algorithm job is illustrated in Fig. 4, where pre includes all the preliminary operations, i.e., the mapping from high-level language formulation to low-level representation, compilation for the target hardware (i.e., transpilation), and selection of a suitable initial set of parameters, while post indicates any postprocessing that might be necessary after all the iterations have been performed. The time and complexity associated with each step vary wildly with the application type and the size of the problem addressed, as well as the classical or quantum hardware where the tasks run. In Section III, we will provide figures obtained from experiments run for a specific optimization application.

B. MULTISERVER PROBLEM FORMULATION

We now propose a standard formulation inspired by the wellknown problem of multiserver scheduling (e.g., [16]), which would align well with the characteristics of a traditional platform for executing hybrid classical–quantum applications, such as that in Fig. 1. Even if this formulation cannot capture

1 Variational quantum computing can be used as well with adiabatic QCs [15], in which case the low-level representation is the Hamiltonian directly describing a solution to the problem of interest. See Section V-A for a discussion on such an alternative.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 5, Issue 1, December 2025 Impact Factor: 7.67

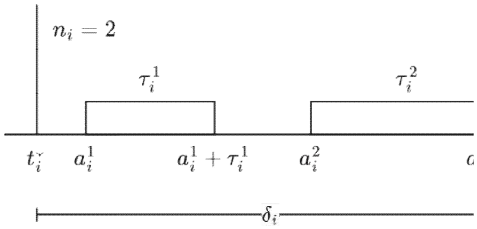


FIGURE 5. Example of a two-task job i, showing the job arrival time ti ↓ , the activation times of the two tasks (a1i and a2i), the task durations (τ i1 and τ i2), and the job delay δ i.

some specific features of a hybrid classical ‐ quantum serverless computing system under study (i.e., Fig. 2), we believe it is useful to introduce the notation and guide the reader. As illustrated in Fig. 5, we assume that the ith job arrives at time ti ↓ and consists of ni tasks (in the example, it is ni = 2), which must be executed sequentially. Each task can be executed either on a classical computer (ci = 1) or on a QC (qi = 1), and the execution of the task requires a duration τi, during which it is not interrupted. The delay of the job δi would then be the time between the job's arrival and when the last task has completed execution. Since the delay directly affects the user's experience, one reasonable objective is to minimize the total delays of all the jobs, i.e.,

$$\min \sum_{i=1}^{N} \delta_i \qquad (1)$$

An interesting case for this is to assume that the resources are finite, i.e., there is a maximum number of classical tasks C that can be executed in parallel, and, similarly, Q is the maximum number of quantum tasks that can run in parallel. Under this assumption, the multiserver classical–quantum problem is to find the activation times aij of any task j of job i such that the constraints in (2) are not violated, which ensures that any task is activated after it starts and that the classical and server capacities are not exceeded.

$$\begin{cases} a1_i \geq t_i^{\downarrow} & \forall i \; aij \geq aiji{-}1 + \tau i \; ij{-}1 \; \forall i \; \forall j \geq 2 \\ & \delta i = ani + \tau in - ti^{\downarrow} \quad \forall i \\ & c_i, q_i \in \{0,1\} \; \forall i, j \\ \sum_{h=1}^{N} \sum_{k=1}^{n_h} \mathbb{I}(2) \; j \; j \; c_i + q_i = 1 \; \forall i, j \\ \sum_{h=1}^{N} \sum_{k=1}^{n_h} \mathbb{I}(i,j,h,k) c_i / c^k{}_k \leq C \quad \forall i, j \\ (i,j,h,k) q_i / q^k \leq Q \; \forall i, j \\ = \qquad {}_k \end{cases}$$

In (2), we use the indicator function I(i, j,h,k), which yields

1 when the two tasks activated at aij and akh are (partially) overlapping in time, i.e., they are executed in parallel

$$\text{hk} \; (3) \begin{cases} 1, & akh \leqslant aij \leqslant akh + \tau hk \vee \\ 0, & \text{otherwise.} \end{cases}$$

formulation would be to minimize, e.g., the classical capacity C under a delay δimax associated with each job i

2 Indeed, we use this feature in Section III.

C. SERVERLESS SYSTEM MODEL

The formulation in the previous section does not capture some key aspects of the hybrid classical–quantum serverless computing system described in Section I and illustrated in Fig. 2.

Observation 1: The formulation in Section II-B assumes that the arrival times ti ↓ of all the jobs are known at decision time, which is typical of offline optimization processes. In a production system, scheduling can be based only on the

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 5, Issue 1, December 2025 Impact Factor: 7.67

currently active jobs, because, in general, there is no way to know about future jobs. In other words, the true problem is an online one.

Observation 2: The duration τij of a task within a job is now known beforehand. In some cases, it can be estimated with some error margin. For instance, IBM Qiskit provides an API to estimate the execution time of a quantum circuit on a specific QC.2 However, the application logic is usually opaque to the serverless platform, because many times the user may ship it in binary format inside a container image, and generally even estimating a serverless task duration is known to be challenging [17].

Observation 3: Container platforms commonly exploit statistical multiplexing of concurrent tasks running in separate containers. In other words, there can be more active containers than CPUs in the system: the CPU capacity may have to be shared between independent applications. Therefore, even if details about the classical application are fully known, it is not possible to know beforehand the execution time as it depends on the time-variable number of active tasks.

Observation 4: Serverless is subject to cold start [18]: the execution time of a task may include a significant amount of time to perform ancillary operations (e.g., load the container image and set up the virtual network), which are needed only if there is no container available that can already host the same task. Again, this affects the capability of the platform to infer the task's execution time based purely on its characteristics.

Observation 5: The number of iterations ni of a variational quantum computing job is not known in advance because it depends on the convergence of the classical optimization of the "update parameters" task in Fig. 3, which in turn depends on the methodology as well as the input data.

Based on these observations, in this work, we propose a simulation model that better captures the distinguishing features of a hybrid classical–quantum serverless computing system. The model is described in the following and was made publicly available as open-source code under a permissive MIT license on GitHub [19], with initial results reported in Section IV.

In our model, the jobs' arrival process follows a Poisson distribution, with rate λ. According to Fig. 4, each job performs classical/quantum iteration tasks preceded by a preparation task and ending with a postprocessing task. Classical tasks are characterized by the number of operations required, whereas quantum tasks are characterized by the execution time. Such a difference allows us to model the different nature of classical versus quantum tasks: the former are executed on containers on a shared pool of CPUs and the latter are executed on dedicated QCs that, with current technology, cannot exploit concurrency and statistical multiplexing [20]. The execution time τij (in seconds) of a classical task is computed as the ratio between the number of operations Nij of that task and the effective operation rate (in number of operations per second). The latter never exceeds the capacity S of a single classical CPU, but can be less than that if there are more active tasks than the total number C of CPUs, in which case we assume that the capacity is spread evenly across the tasks. Therefore, it is

$$\tau i = S \cdot \min\{1, C/A\} \quad (5)$$

assuming that the number of active tasks remains unchanged until the task's end; otherwise, it is adjusted accordingly. Each job is also assigned a priority to differentiate jobs. For example, this can be used to give fewer resources to jobs from users who have used up all their quotas or to match different pricing profiles. In the next section, we discuss our methodology to create a dataset from which to randomly draw the number of iterations, the number of operations of classical tasks, and the execution time of quantum tasks.

Furthermore, a scheduler is needed to assign online the quantum tasks to the Q available QCs. In this work, we consider the following four well-known policies, whose impact on the performance is assessed in Section IV. 1) FIFO: Schedule the oldest task.

2) LIFO: Schedule the newest task.

3) Random: Schedule a random task with even probability.

4) Weighted: Schedule a random task, where each task is assigned a probability to be selected proportional to the task's priority.

Finally, to prevent the system from becoming unstable, i.e., queueing times growing arbitrarily, we perform a basic admission control on the incoming jobs based on the current status of the system. A new job is accepted only if the number of classical tasks currently active does not exceed Cmax and the number of quantum tasks, pending and in

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 1, December 2025
Impact Factor: 7.67

execution, does not exceed Qmax. After a job is accepted by the platform, it is never interrupted to avoid wasting resources for partial computations performed.

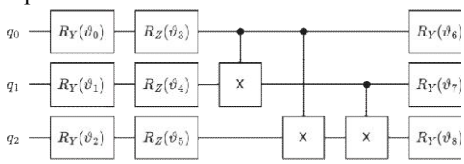


FIGURE 6. Example: 3-qubit quantum circuit used in the Qiskit serverless experiments.

The full set of parameters available in the simulator is reported in the Nomenclature.

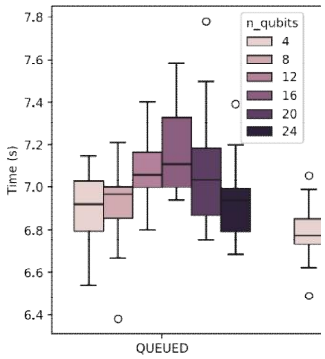
III. QISKIT SERVERLESS EXPERIMENTS

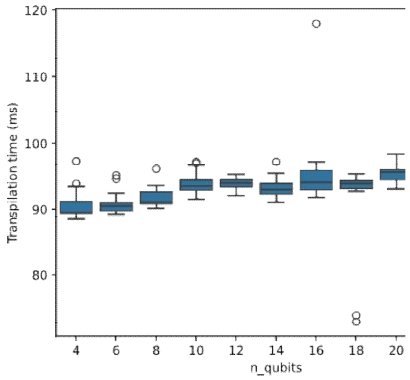
This section describes the methodology adopted to create a dataset that will be used as input for the simulations in Section IV. We deployed a local serverless computing platform following the instructions of the IBM Qiskit serverless platform [21], which are provided to enable researchers to replicate on their infrastructure a platform for developing and testing their hybrid serverless computing applications without accessing the IBM cloud production systems. Such a platform uses Ray [22], which is a framework for the con- current execution of artificial intelligence jobs, optimized for programs written in the Python programming language, also commonly adopted for quantum computing applications, including in Qiskit. We deployed the cluster through the Docker Compose tool [23], which automatizes creating and managing multiple interacting Docker containers, but other options are possible, including the use of Kubernetes [24] and public cloud services. We used a high-end server for the local installation, with Intel Xeon Silver 4410 T CPU and 1 TB of RAM.

We selected a target example application from HamLib [25], a library of qubit-based quantum Hamiltonians intended for benchmarking quantum systems. In particular, we selected the MaxCut problem, with random three-regular graphs, for which 20 instances are provided for quantum circuits ranging from 4 to 90 qubits.

We then developed the corresponding QAOA application following a serverless pattern using the Qiskit library, also saving timestamps on the occurrence of noteworthy job events. In the client, running on the user end, we monitor the time the job remains in a queued versus initialization status. On the other hand, the function itself, executed by the platform, determines the execution times of the classical and quantum iteration tasks. The quantum circuit used includes rotation and cnot gates and was generated with the Qiskit function EfficientSU2. For example, for three qubits, the quantum circuit is reported in Fig. 6 and has 12 parameters(ϑi)that are initialized with random values and optimized by a classical task at each new iteration.

In Table 1, we report the correspondence between all the metrics plotted in this section and the identifiers used in the datasets.





IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 1, December 2025
Impact Factor: 7.67

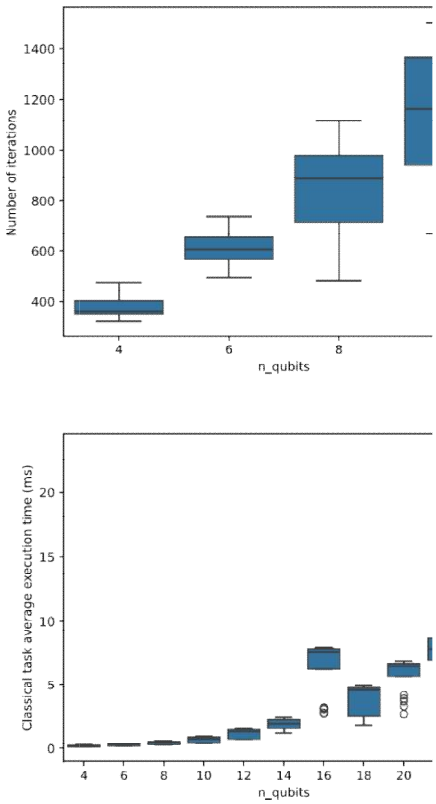


FIGURE 7. Local Qiskit serverless experiments, for problems instances with different sizes. (Top left) Time spent by a job while queueing and during initialization. (Top right) Number of iterations needed (only until ten qubits, maximum number of iterations set to 1500). (Bottom left) Duration of the transpilation of the quantum circuit. Bottom right: duration of a classical iteration.

Plot	Dataset	Column(s)
Figure 7 TL	output_single	QUEUED, INIT
Figure 7 TR	output_single	num_iterati
Figure 7 BL	output_single	run_transpi
Figure 7 BR	output_single	avg_clas_it
Figure 8	output_series	cost
Figure 9 L	output_series	time
Figure 9 R	ibm_job_estimate	4th column

In the Plot column: T = top, B = bottom, L = left, R = right.

TABLE 1. Correspondence Between the Values in Section III's Plots and the Respective DATASETS/COLUMNS in the Repository [19]

In Fig. 7, we report the results obtained with our local Qiskit serverless platform, with problem instances corresponding to quantum circuits from 4 to 24 qubits. With bigger instances, the time to simulate quantum computing tasks becomes unacceptable. In the top left plot, we show the time during which the job appears with status QUEUED and INITIALIZING in the platform, which is needed for the upload and initialization of the job's function and the setup of the run-time environment, including loading Python libraries. As can be seen, such times are both independent of the problem size. Since our local testbed was unused except for the current experiment, and jobs were created back to back (no tin parallel),there was no additional queueing due to concurrent/interfering computing processes. The other main classical tasks performed are the transpilation of the quantum circuit, which is performed only once and is reported in the bottom left plot, and the operations related to solving the classical minimization, performed at each iteration and reported in the bottom right plot. The latter depends on the specific algorithm adopted; in our experiments, we used constrained optimization by linear approximation, provided by the SciPy library [26]. Both transpilation and optimization increase with the problem size, in particular, the latter rather steeply, but remains small compared to the

DOI: 10.48175/IJARSCT-30155

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 1, December 2025
Impact Factor: 7.67

duration of the quantum task at each iteration, both in a QC and simulated in a classical computer, as we show later. Finally, in the top right plot, we show the number of iterations, which increases significantly with the number of qubits; in fact, to keep the overall duration of the experiments feasible, we set a hard limit to 1500 iterations, which is also justified by the following observation.

In Fig. 8, we show the effectiveness of the optimization method used, measured as the relative difference between the initial and final value of the cost function: the initial value is given by a random configuration of the problem variables, while the final value corresponds to the minimum energy state found, unless the algorithm terminated early due to exceeding the maximum number of iterations. As can be seen, with four and six qubits, the solver imost effective,

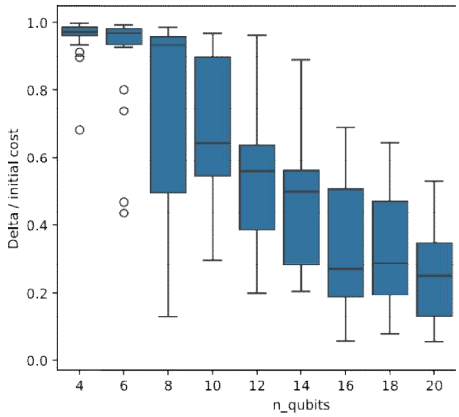


FIGURE 8. Local Qiskit serverless experiments, for problems instances with different sizes: effectiveness of the variational quantum computing algorithm, in terms of the difference between the initial and final cost function value, normalized by the initial cost.

as the metric is close to 1. However, it rapidly decreases even with eight and ten qubits, despite convergence being reached before the maximum number of iterations is hit (see again the top right plot in Fig. 7). Slow (or no) convergence is a known issue of variational quantum computing, which becomes especially relevant with current noisy Q Chardware. Even if solutions have been proposed for specific use cases, e.g., [27], well-defined general best practices are not yet available and go beyond the scope of this work.

In the left plot of Fig. 9, we show the execution time of the quantum task at each iteration, simulated on a classical computer. As expected, the time increases exponentially with the quantum circuit size (note that the y-axis has a logarithmic scale), due to a corresponding exponential increase of the state space to be managed. On the other hand, in the right plot, we show the estimated time to execute the quantum circuits of our test application, from 4 to 90 qubits, as reported by the IBM Quantum platform for three different backends with heterogeneous hardware characteristics. Executing small quantum circuits on today's quantum hardware remains inefficient, as they can be simulated exactly on standard classical hardware in a fraction of the time. Notably, and as confirmed by the right plot in Fig. 9, the execution time of a quantum circuit on current devices depends only marginally on the circuit's size. This characteristic opens the door to using quantum hardware for problem instances that are intractable for classical algorithms. However, realizing this potential hingeson over coming significant research challenges, particularly in noise mitigation and the development of logical qubits [28].

By performing a selection of Max Cut experiments on IBM Quantum, we confirmed that the estimations returned provide an accurate representation of the actual time needed to execute the quantum tasks. Queueing times, instead, are wild and unpredictable since they depend on the number and types of jobs pending execution on IBM Quantum, created

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 5, Issue 1, December 2025 Impact Factor: 7.67

Parameter	Small	Big
C	1	$\{1,\ldots,6\}$
S	10^9	10^7 or 10^9
Q	4	$\{1,\ldots,6\}$
$C_{\max}$	40	40
$Q_{\max}$	40	40
Interarrival	600 s	$\{150,\ldots,900\}$ s
Num. qubits	$U[\{4,6,8,10\}]$	$U[\{4,8,12,16,20,24\}]$
Priority	$U[\{1,2,4\}]$	*same*
Policy	*all*	R
Duration	7 days	7 days
Warm-up	12 hours	12 hours
Repetitions	100	100

TABLE 2. Configuration of the Two Simulation Scenarios

by other users worldwide, as well as their respective credits/priorities and how they are managed by the inscrutable IBM scheduling system. Full execution of the experiments on IBM Quantum was not possible due to the monetary costs associated with the use of the platform. The dataset we created with our local infrastructure and emulated classical computers required the execution of 258880 quantum circuits, corresponding to about 600 h of net QC usage. However, the usage billed by IBM, called "actual usage," also includes QC overheads and is subject to rounding effects because of a minimal quantum processing unit granularity; experience suggests that with these type of experiments, it is about 7× bigger than the net usage, which would yield a rough estimate of 4200 h billed, costing up to US$ 12 million with a Premium Plan [29].

IV. SIMULATION EXPERIMENTS

In this section, we analyze the results obtained with a numerical simulator developed based on the model in Section IIC, which is available as open-source in GitHub [19], as already mentioned, also including the scripts to perform all the experiments in this section and to plot the results.

We carried out two simulation campaigns, whose parameters are reported in Table 2.

1) Small uses the execution times obtained with the simulation of a Q Conclassical hardware(leftplotinFig.9), normalized on the execution times estimated by IBM Quantum (right plot in Fig. 9) by choosing the best executor selected by the platform at the time when the simulations have been performed. The name "small" is because we focus on how the different scheduling policies in Section II-C, i.e., FIFO (F), LIFO (L), Random (R), and Weighted (W), affect the performance of multipriority jobs. Indeed, each job entering the system is assigned a random problem instance, from 4 to 10 qubits, and a random priority drawn in {1,2,4}.

2) Big focuses on how the provisioning of the system, in terms of the number C of classical serverless workers (and their computation capacity S) and the number Q of QCs, affect the performance of the system, when the load of average jobs entering the system varies. When studying the provisioning of classical serverless

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 1, December 2025
Impact Factor: 7.67

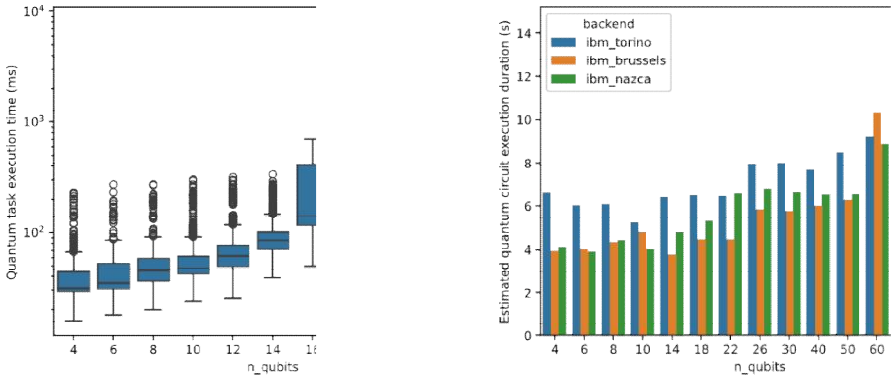


FIGURE 9. (Left) Duration of a quantum iteration simulated on a classical computer in local Qiskit serverless experiments.
(Right) Estimation of the time needed for a single execution of the quantum circuit in our test application, provided by IBM Quantum for different backends: ibm_torino (Heron r1, 30 k Circuit Layer Operations Per Second (CLOP)), ibm_brussels (Eagle r3, 37 k CLOPS), and ibm_nazca (Eagle r3, 29 k CLOPS).

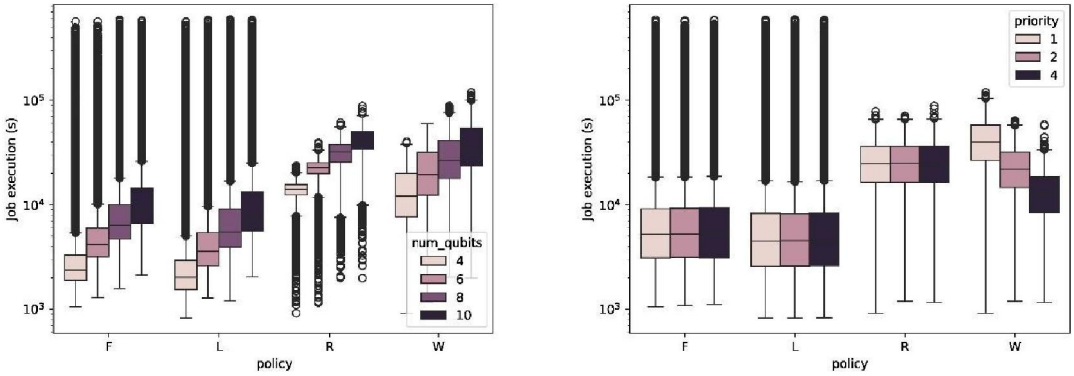


FIGURE 10. Small scenario. Job execution time with different scheduling policies and grouped (left) per number of qubits and (right) priority. workers, we reduced the worker capacity from 109 to 107 operations/s. For simplicity of analysis, we use homogeneous priority in this scenario and focus only on the Random (R) scheduling policy. The name "big" stresses that jobs use instances with up to 24 qubits.
Each scenario has been repeated 100 times, following a Monte Carlo methodology. Most of the results are plotted with box-and-whisker plots, showing the 25%, 50%, and 75% quartiles in the box, with whiskers extending to the overall population of values, except for outliers plotted individually. We consider the following performance metrics.
1) Job execution time: The time between when a job enters the platform and when it is completely dispatched, i.e., all its tasks have been executed fully and in order.
2) Average classical tasks: Average number of active jobs currently executing a classical task.
3) Average quantum tasks: Average length of the queue of quantum tasks for active jobs.
4) Drop probability: The number of jobs that are denied execution in the platform due to the currently active classical tasks exceeding Cmax or the number of pending quantum tasks exceeding Qmax, divided by the total number of jobs that requested to be admitted.

A. SMALL SCENARIO

In the "small" scenario, we compare the scheduling policies (in the x-axis in the plots) and focus on the job execution time as the key performance metric, as reported in Fig. 10. The drop probability (not shown) is comparable for all the scheduling policies, with an average of 15%. In the left plot, the job execution time is grouped by the number of qubits.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 5, Issue 1, December 2025 Impact Factor: 7.67

It increases with the problem size, mostly because of the correlation with the number of iterations in the dataset (top right plot in Fig. 7). We observe that F and L policies both yield a significantly lower job execution time in the boxvalues, i.e., those in 25%–75% quartiles interval, than random policies R and W. This can be explained as follows. With F, i.e., FIFO, once a job has its quantum task enqueued, no other jobs (new or preexisting) can overrun it, which ensures that each task has to wait in queue for the full duration of its traversal: this favors jobs with fewer iterations. The situation is flipped with L, i.e., LIFO, where new quantum tasks entering the queue are always preferred to older ones: jobs with a shorter iteration duration will be favored in the long term because they will visit the queue more frequently than others with longer iteration execution times. Because of the bias of both policies toward one job type or another, there is a high number of outliers with a very high job execution time. On the other hand, the randomized policies R and W exhibit substantially lower tail job execution times because they do not suffer from biases, as can be seen by the smaller population of outliers. However, this comes at the expense of higher job execution time for most jobs.

In the right part of Fig. 10, we show the same results grouped by the job priority. The F, L, and R policies do not consider this property of the jobs, and the job execution times are the same for all the priorities. Instead, we designed W to be priority-aware, since it schedules quantum tasks with a probability proportional to the job's priority value. The results prove this simple mechanism to be effective, as the job execution time is proportional to the respective priority of jobs. Take-away messages: Scheduling of quantum tasks in iterative variational quantum computing applications running in a hybrid classical–quantum serverless infrastructure can significantly affect the performance. In particular, randomized policies increase the average job execution times while reducing tail values. Furthermore, we note that a simple weighted randomized policy effectively differentiates jobs with mixed priorities.

A. BIG SCENARIO

In this section, we illustrate the results obtained in the so called "big" scenario, which differs from the previous one in the size of problem instances, which now go from 4 to 24 qubits. Furthermore, for simplicity of analysis, we assign the same priority to all jobs and use a single policy (Random, which provides lower tail job execution times compared to sorted alternatives), which allows us to assess the performance with variable loads and resources.

First, we vary the number of QCs (Q), with a serverless worker capacity that creates a bottleneck on the quantum resources, i.e., S = 109. In Fig. 11, we show the drop probability as a heat map to visualize at the same time the impact of both Q and the load. As expected, the drop probability decreases steadily when increasing Q for a given load. In this respect, the simulation model can be used as a tool to provision there sources of a hybrid classical–quantum serverless system for jobs with given statistical characteristics or following a trace-driven approach.

We now move to the case when classical resources are scarce, by decreasing the serverless worker capacity to S = 107. In Fig. 12, we show a heat map of the drop probability with varying load and number of serverless classical workers (C). The drop probability increases with an increasing load,

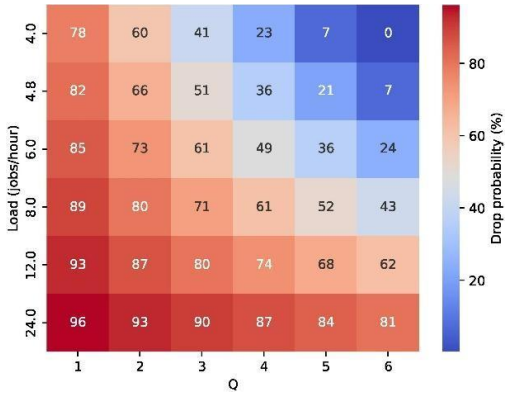


FIGURE 11. Big scenario. Drop probability when varying the number of QCs (Q) with different load conditions.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 1, December 2025
Impact Factor: 7.67

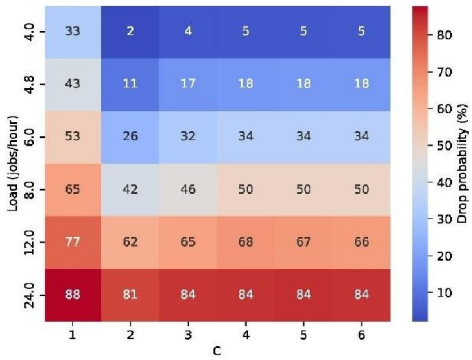


FIGURE 12. Big scenario. Drop probability when varying the number of serverless classical workers (C) with different load conditions.

as expected. However, when looking at the same metric with constant load, the behavior is nonmonotonic: the drop probability is maximum with C = 1, then decreases sharply to a minimum and increases again when adding further serverless classical workers. This phenomenon can be explained by delving into the internals of the simulated platform. In Fig. 13, we show the average number of tasks, both classical (left) and quantum (right). Recall that, as explained in Section II-C, we drop a new job if there are too many classical or quantum tasks already active in the system, with Cmax = Qmax = 40 in this scenario. As C increases, the availability of classical resources grows, which reduces the average number of classical tasks currently being served but also increases the average number of active quantum tasks. Such an interplay between classical and quantum resources is inherent to the hybrid infrastructure under study and leads to the nonmonotonic trend in Fig. 12. Finally, we show in Fig. 14 the execution time for a representative load (8 jobs/h). With C = 1, the performance is worst, although not by a large amount, due to the shaping of incoming jobs, which keeps the system stable even at high loads. With C = 2, for which we have the smallest drop probability (see Fig. 12), the job execution time reduces significantly, although a nonnegligible amount

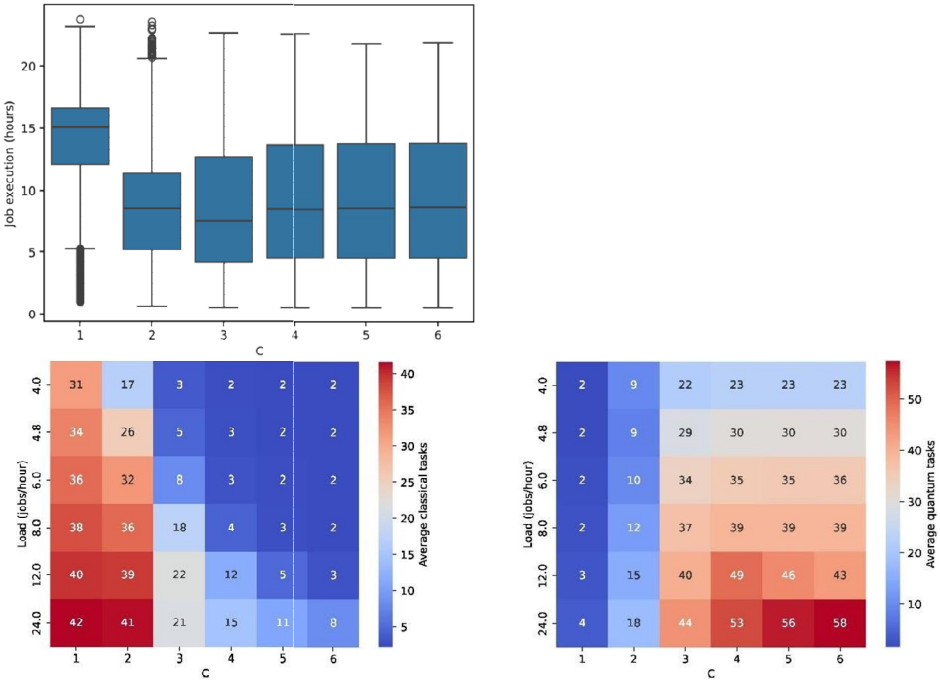


FIGURE 13. Big scenario. (Left) Average classical versus (right) quantum tasks, when varying the number of serverless classical workers (C) with different load conditions. It is Cmax = Qmax = 40. Based on how the admission

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 5, Issue 1, December 2025 Impact Factor: 7.67

control is defined in Section II-C, the average number of active classical or quantum tasks may exceed the respective limit, but the sum of classical and quantum tasks can never exceed Cmax + Qmax.

FIGURE 14. Big scenario. Job execution time when varying the number of serverless classical workers (C) with a load of 8 jobs/h. of outliers remains on the upper bound. As the number of classical serverless workers increases further (C > 2), the 25% and 50% percentiles of the job execution time (middle line in boxes) initially decrease then increase again, while the75%percentile increases teadily;in any case, the outliers disappear. This is because the interplay between classical and quantum resources becomes progressively less relevant as C increases until the system dynamics are dominated by the queueing of quantum tasks only.

Take-away messages: Depending on the load characteristics and availability of resources, there can be an interplay between classical and quantum tasks leading to performance trends nonmonotonic with the resources deployed, in terms of the drop probability and job execution time. The simulation model defined can capture these effects and be used as a tool to provision a hybrid classical–quantum serverless platform.

V. DISCUSSION

In this section, we discuss the limitations of the current work (see Section V-A) and report some lessons learned during the execution of the related research activities (see Section V-B).

A. LIMITATIONS

While the simulation model in Section II-C is generic, some of the conclusions in this work may depend specifically on the dataset used as input. As described in Section III, such a dataset has been derived based on experiments run in a single platform, i.e., IBM Qiskit serverless, which use Ray, in a local computing infrastructure, complemented by quantum circuit job execution estimated by the IBM Quantum system. A different combination of serverless platforms and QCs would likely give different results, which might affect, at least quantitatively, some of the conclusions in this article. Furthermore, the dataset has been determined using a single application, i.e., MaxCut problems with random three-regular graphs: even if this can be considered representative of a broad class of variational quantum computing algorithms, it is an open question how much the simulation model and conclusions remain valid with different applications (VQE, QAOA, or others). Furthermore, as briefly introduced earlier, a broad range of variational quantum computing algorithms can be realized with analog QCs, such as those manufactured by D-Wave [30] with up to 5000 qubits. With today's technology, the latter supports many more qubits than gate-based QCs, which can change the balance between the complexity of classical versus quantum tasks. As part of our future work, we will extend our research activities to include data from the execution in analog QCs provided by Pasqal [31] to the project HPCQC [32], funded by the European Commission in the EuroHPC framework.

Concerning the simulation model, it is very far from the level of maturity and complexity of modern serverless computing simulators (e.g., [33]). For instance, we model in a simplistic manner the effects of cold-start, while neglecting the container platform and scheduling overhead, and the costs incurred by data transfer, which in cloud computing production systems have been shown to impact the performance significantly [34]. This is because this work aims at providing an initial understanding of the interplay between classical and quantum tasks for a specific class of applications, i.e., variational quantum computing, with serverless computing. Further research will be needed, backed up by data acquired in realistic/production environments, to better understand which factors from classical serverless computing are worth incorporating into the system model and which new aspects need to be prioritized for performance assessment purposes.

B. LESSONS LEARNED (ABOUT IBM QISKIT SERVERLESS)

As it happens frequently, when performing experiments, we encountered some findings about the tools used, which we share with the community in the hope that they will streamline future research activities in this area. First, deploying a local Ray cluster as a set of Docker containers following the instructions provided by IBM Qiskit serverless (version 0.17.1) has been very straightforward and worked out of the box. However, documentation was a bit scarce on the customization of the setup, including for the deployment of a "development" flavor with full monitoring, which

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 5, Issue 1, December 2025 Impact Factor: 7.67

required acquiring familiarity with the individual tools used, whose roles and connections had to be inferred from Docker Compose configuration files and external online resources.

Second, the local infrastructure allowed the development of the test application without resorting to using IBM Quantum resources, which has been immensely useful and reduced substantially the effort and pain associated with the application development. However, the local infrastructure is not pin-compatible with IBM Quantum. On the one hand, there is a mismatch in the versions of the libraries used in the local infrastructure versus IBM Quantum, which has led to difficult-to-troubleshoot issues. For instance, the constructor of the Session Python class had a different syntax, which caused a run-time exception from within the serverless worker; such failures are known to be difficult to debug also in mature and production-ready classical serverless platforms. On the other hand, the local infrastructure only implements a fraction of the IBM Quantum modes of operation and user management schemes, which required a frustrating trial-and-error process to fully deploy the application.

Finally, IBM Quantum does not yet include a mode dedicated to serverless. Currently, the supported modes are job, batch, and session [35]. Job and batch modes only allow quantum computing operations, with the only difference being that in batch mode, it is possible to lock the QC for multiple operations. Whether or not this is beneficial depends on cases, and the documentation provides some rules of thumb. The session mode allows alternating between quantum and classical operations and is intended for applications, such as variational quantum computing. The details on how this mode is handled in the backend are not disclosed by IBM, but the documentation hints that quantum operations in a batch can be prioritized to reduce the overall service time. The FaaS programming model of serverless computing assumes that functions are stateless elementary entities; therefore, using the batch or session mode would invalidate the underlying assumption. However, the job mode is associated with the highest platform overhead and, in fact, is explicitly only intended for short and sporadic executions.

Overall, IBM Qiskit serverless is a promising solution, which however appears to be far less mature than other streamlined options for the development and execution of applications in IBM Quantum. By looking at the history of serverless computing in cloud environments, we believe that this pattern will eventually emerge as a dominating one, not only at IBM, because of its benefits: flexibility, low development barrier, and billing granularity, among the others. However, there are still many aspects that have to be studied in the specific context of quantum applications, which are not fully covered by existing tools and prior literature, as surveyed in the next section.

VI. RELATED WORK

The use of serverless platforms for quantum computing is not new in the scientific literature. Grossi et al. [36] proposed a framework to integrate cloud quantum computing services with a local serverless platform using Apache Kafka as the underlying mechanism for function invocation. A prototype implementation has been tested with IBM QCs resulting in a collection of lessons learned and best practices for future implementations. Another framework ,called QFaaS, was put forward by Nguyen et al. [37], where they focused on software engineering aspects and also provided tools for end users, including a dashboard for monitoring the running jobs and a command-line interface to support automation and scripting. In this article, the authors showcase quantum random number generation and the execution of Shor's algorithm on IBM QCs. Finally, Stirbu et al. [38] studied the mapping between quantum resources and artifacts to the concepts used in Kubernetes [24], a popular container management platform used in many serverless computing environments. None of the serverless platforms in the scientific literature have been designed specifically or optimized for executing variational quantum circuits.

Some initial studies have dealt with the modeling and optimization of quantum computing resource scheduling. Kaewpuang et al. [39] jointly optimized the time to transpile and execute quantum circuits on heterogeneous QCs, which, however, requires full knowledge of the quantum application and is not designed for iterative tasks, such as in variational quantum computing. Alvarado-Valiente et al. [40] focused more specifically on the opportunity to load balance between multiple quantum computing service providers and they provide a software architecture and APIs for this purpose; they tested their findings with IBM Quantum and Amazon Braket QCs, showing the potential advantages for both developers and users. A framework for benchmarking a hybrid classical–quantum platform was presented by Karalekas et al. [41], where they also proposed optimizations for variational quantum computing algorithms tested in

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 5, Issue 1, December 2025 Impact Factor: 7.67

the Rigetti quantum cloud services platform resulting in significant latency improvements. Finally, concerning the tools for the simulation of hybrid classical–quantum systems, we mention iQuantum [42], which however does not model serverless computing platforms.

We note that serverless computing is not the only option under study to improve the scalability and usability of hybrid classical–quantum infrastructures. For instance, Marosi et al. [43] proposed a reference architecture relying on traditional (nonserverless) principles that allow the seamless use of cloud resources, in virtual machines or containers, through a gateway providing users with the ability to interact with JupyterLab notebooks [44], which is particularly appealing as a learning platform. On the other hand, Claudino et al. [45] showed how to use the XACC compilation framework [46] to massively parallelize the simulation of quantum circuits on classical hardware using NVIDIA's cuQuantum [47]. Finally, a pure HPC approach has been adopted at the Oak Ridge National Laboratory, as described in [48], to integrate heterogeneous classical and quantum computing hardware to support scientific applications, such as quantum manybody dynamics and continuum mechanics simulations, in addition to more common optimization and QML.

VIII. CONCLUSION

In this article, we have studied the opportunity of adopting serverless computing to execute variational quantum computing algorithms in a hybrid classical–quantum infrastructure. We have defined a system model emerging from observations on the differences between a traditional queueing based system and serverless computing, which has led to the development of a numerical simulator, which was made available as open-source to the community. To feed the simulator, we performed experiments in a local infrastructure deployed according to the IBM Qiskit instructions, with the use case of an optimization MaxCut application from the public HamLib benchmarks, where the execution time of quantum circuits was estimated on different backends at IBM Quantum. This has allowed us to derive a few lessons we shared. A simulation study has been carried out and showcased the most relevant system features, including how the choice of quantum task scheduling affects the performance, with four alternatives compared, the opportunity of assigning priority to jobs, and the aspects related to the provisioning of classical versus quantum resources, which depends on a nontrivial interplay between the two. The limitations of our approach have been outlined, together with a sketch of possible open directions.

REFERENCES

[1] S. S. Gill and R. Buyya, "Transforming research with quantum computing," J. Economy Technol., Jul. 2024, Art. no. S2949948824000295, doi: 10.1016/j.ject.2024.07.001.

[2] F. Bova, A. Goldfarb, and R. G. Melko, "Commercial applications of quantum computing," EPJ Quantum Technol., vol. 8, no. 1, Dec. 2021, Art. no. 2, doi: 10.1140/epjqt/s40507-021-00091-1.

[3] M. Coccia, S. Roshani, and M. Mosleh, "Evolution of quantum computing: Theoretical and innovation management implications for emerging quantum industry," IEEE Trans. Eng. Manage., vol. 71, pp. 2270–2280, 2024, doi: 10.1109/TEM.2022.3175633.

[4] E. G. Rieffel et al., "Assessing and advancing the potential of quantum computing: A NASA case study," Future Gener. Comput. Syst., vol. 160, pp. 598–618, Jun. 2024, doi: 10.1016/j.future.2024. 06.012.

[5] H. Baniata, "SoK: Quantum computing methods for machine learning optimization," Quantum Mach. Intell., vol. 6, no. 2, Dec. 2024, Art. no. 47, doi: 10.1007/s42484-024-00180-1.

[6] M. E. S. Morales, J. D. Biamonte, and Z. Zimborás, "On the universality of the quantum approximate optimization algorithm," Quantum Inf. Process., vol. 19, no. 9, Sep. 2020, Art. no. 291, doi: 10.1007/s11128-02002748-9.

[7] A. Peruzzo et al., "A variational eigenvalue solver on a photonic quantum processor," Nature Commun., vol. 5, no. 1, Jul. 2014, Art. no. 4213, doi: 10.1038/ncomms5213.

[8] T. S. Humble, A. McCaskey, D. I. Lyakh, M. Gowrishankar, A. Frisch, and T. Monz, "Quantum computers for high-performance computing," IEEE Micro, vol. 41, no. 5, pp. 15–23, Sep./Oct. 2021, doi: 10.1109/ MM.2021.3099140.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 5, Issue 1, December 2025 Impact Factor: 7.67

[9] S. Ali, T. Yue, and R. Abreu, "When software engineering meets quantum computing," Commun. ACM, vol. 64, no. 4, pp. 84–88, Apr. 2022, doi: 10.1145/3512340.
[10] S. Kounev et al., "Serverless computing: What it is, and what it is not?," Commun. ACM, vol. 66, no. 9, pp. 80–92, 2023, doi: 10.1145/3587249.
[11] K. Rzadca et al., "Autopilot: Workload autoscaling at Google," Proc. 15th Eur. Conf. Comput. Syst., EuroSys 2020, 2020, Art. no. 16, doi: 10.1145/3342195.3387524.
[12] Y. Li, Y. Lin, Y. Wang, K. Ye, and C.-Z. Xu, "Serverless computing: State-of-the-art, challenges and opportunities," IEEE Trans. Services Comput., vol. 16, no. 2, pp. 1522–1539, Mar./Apr. 2022, doi: 10.1109/ TSC.2022.3166553.
[13] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik, "The theory of variational hybrid quantumclassical algorithms," New J. Phys., vol. 18, no. 2, Feb. 2016, Art. no. 023023, doi: 10.1088/13672630/18/2/023023.
[14] O. Ezratty, "Perspective on superconducting qubit quantum computing," Eur. Phys. J. A, vol. 59, no. 5, May 2023, Art. no. 94, doi: 10.1140/ epja/s10050-023-01006-7.
[15] 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, doi: 10.1088/13616633/ab85b8.
[16] I. Grosof, M. Harchol-Balter, and A. Scheller-Wolf, "WCFS: A new framework for analyzing multiserver systems," Queueing Syst., vol. 102, no. 1–2, pp. 143–174, Oct. 2022, doi: 10.1007/s11134-022-09848-6.
[17] C. Lin, N. Mahmoudi, C. Fan, and H. Khazaei, "Fine-grained performance and cost modeling and optimization for FaaS applications," IEEE Trans. Parallel Distrib. Syst., vol. 34, no. 1, pp. 180–194, Jan. 2023, doi: 10.1109/TPDS.2022.3214783.
[18] H.Yuetal.,"RainbowCake:Mitigatingcold-startsinserverlesswithlayerwise container caching and sharing," in Proc. 29th ACM Int. Conf. Architectural Support Program. Lang. Operating Syst., La Jolla, CA, USA, Apr. 2024, vol. 1, pp. 335–350, doi: 10.1145/3617232.3624871.
[19] "Paper code repository, tag v1.0.0," Accessed: 15 Apr. 2025. [Online]. Available: https://github.com/ccicconetti/serverless_quantum_sim
[20] S. Niu and A. Todri-Sanial, "Enabling multi-programming mechanism for quantum computing in the NISQ era," Quantum, vol. 7, Feb. 2023, Art. no. 925, doi: 10.22331/q-2023-02-16-925.
[21] "Qiskit serverless," Accessed: 15 Apr. 2025. [Online]. Available: https://qiskit.github.io/qiskit-serverless/
[22] "Ray," Accessed: 15 Apr. 2025. [Online]. Available: https://www.ray.io/
[23] "Docker composer," Accessed: 15 Apr. 2025. [Online]. Available: https://docs.docker.com/compose/
[24] "Kubernetes," Accessed: 15 Apr. 2025. [Online]. Available: https://kubernetes.io/
[25] N. P. Sawaya et al., "HamLib: A library of Hamiltonians for benchmarking quantum algorithms and hardware," in Proc. IEEE Int. Conf. Quantum Comput. Eng., Bellevue, WA, USA, Sep. 2023, pp. 389–390, doi: 10.1109/QCE57702.2023.10296.
[26] "SciPy," Accessed: 15 Apr. 2025. [Online]. Available: https://scipy.org/ [27] F. Arute et al., "Hartree-fock on a superconducting qubit quantum computer," Science, vol. 369, no. 6507, pp. 1084–1089, Aug. 2020, doi: 10.1126/science.abb9811.
[28] Google Quantum AI Consortia, "Suppressing quantum errors by scaling a surface code logical qubit," Nature, vol. 614, no. 7949, pp. 676–681, Feb. 2023, doi: 10.1038/s41586-022-05434-1.
[29] "IBM quantum computing pricing," Accessed: 15 Apr. 2025. [Online]. Available: https://www.ibm.com/quantum/pricing
[30] "D-Wave Quantum Inc.," Accessed: 15 Apr. 2025. [Online]. Available: https://www.dwavesys.com/
[31] "SAS Pasqal," Accessed: 15 Apr. 2025. [Online]. Available: https://www.pasqal.com/
[32] "High Performance Computer–Quantum Simulator hybrid (HPCQS)," Accessed: 15 Apr. 2025. [Online]. Available: https://hpcqs.eu/

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 5, Issue 1, December 2025 Impact Factor: 7.67

[33] A. Matricardi, A. Bocci, S. Forti, and A. Brogi, "Simulating FaaS orchestrations in the cloud-edge continuum," in Proc. 3rd Workshop Flexible Resource Appl. Manage. Edge, Orlando, FL, USA, Aug. 2023, pp. 19–26, doi: 10.1145/3589010.3594893.
[34] Q. Liu, D. Du, Y. Xia, P. Zhang, and H. Chen, "The gap between serverless research and real-world systems," in Proc. ACM Symp. Cloud Comput., Santa Cruz, CA, USA, Oct. 2023, pp. 475–485, doi: 10.1145/3620678.3624785.
[35] "Introduction to Qiskit runtime execution modes," Accessed: 15 Apr. 2025. [Online]. Available: https://docs.quantum.ibm.com/guides/ execution-modes
[36] M. Grossi et al., "A serverless cloud integration for quantum computing," Jul. 2021, arXiv:2107.02007, doi: 10.48550/arXiv.2107.02007.
[37] H. T. Nguyen, M. Usman, and R. Buyya, "QFaaS: A serverless function-as-a-service framework for quantum computing," Future Gener. Comput. Syst., vol. 154, pp. 281–300, May 2024, , doi: 10.1016/j.future.2024.01.018.
[38] V. Stirbu, O. Kinanen, M. Haghparast, and T. Mikkonen, "Qubernetes: Towards a unified cloud-native execution platform for hybrid classic-quantum computing," Inf. Softw. Technol., vol. 175, Nov. 2024, Art. no. 107529, doi: 10.1016/j.infsof.2024.107529.
[39] R. Kaewpuang, M. Xu, D. Niyato, H. Yu, Z. Xiong, and J. Kang, "Stochastic qubit resource allocation for quantum cloud computing," in Proc. NOMS 2023-2023 IEEE/IFIP Netw. Operations Manage. Symp., May 2023, pp. 1–5, doi: 10.1109/NOMS56928.2023.10154430, iSSN: 2374-9709.
[40] J. Alvarado-Valiente, J. Romero-Álvarez, E. Moguel, J. García-Alonso, and J. M. Murillo, "Orchestration for quantum services: The power of load balancing across multiple service providers," Sci. Comput. Program., vol. 237, Oct. 2024, Art. no. 103139, doi: 10.1016/j.scico.2024.103139.
[41] P. J. Karalekas, N. A. Tezak, E. C. Peterson, C. A. Ryan, M. P. da Silva, and R. S. Smith, "A quantum-classical cloud platform optimized for variational hybrid algorithms," Quantum Sci. Technol., vol. 5, no. 2, Apr. 2020, Art. no. 024003, doi: 10.1088/2058-9565/ab7559.
[42] E. Moguel, J. Rojo, D. Valencia, J. Berrocal, J. Garcia-Alonso, and J. M. Murillo, "Quantum service-oriented computing: Current landscape and challenges," Softw. Qual. J., vol. 30, pp. 983–1002, Apr. 2022, doi: 10.1007/s11219-022-09589-y.
[43] A. C. Marosi, A. Farkas, T. Máray, and R. Lovas, "Towards a quantum - science gateway: A hybrid reference architecture facilitating quantum computing capabilities for cloud utilization," IEEE Access, vol. 11, pp. 143913–143924, 2023, doi: 10.1109/ACCESS.2023.3342749
[44] "Jupyterlab," Accessed: 15 Apr. 2025. [Online]. Available: https://jupyter.org/
[45] D. Claudino, D. I. Lyakh, and A. J. McCaskey, "Parallel quantum computing simulations via quantum accelerator platform virtualization," Future Gener. Comput. Syst., vol. 160, Jun. 2024, Art. no. S0167739X24003054, doi: 10.1016/j.future.2024.06.007.
[46] "XACC," Accessed: 15 Apr. 2025. [Online]. Available: https://xacc. readthedocs.io/
[47] H. Bayraktar et al., "cuQuantum SDK: A high-performance library for accelerating quantum science," in Proc. IEEE Int. Conf. Quantum Comput. Eng., Bellevue, WA, USA, Sep. 2023, pp. 1050–1061, doi: 10.1109/QCE57702.2023.00119.
[48] T. Beck et al., "Integrating quantum computing resources into scientific HPC ecosystems," Future Gener. Computer Syst., vol. 161, pp. 11–25, Jul. 2024, doi: 10.1016/j.future.2024.06.058.