Trials and Tribulations of Developing Hybrid Quantum-Classical Microservices Systems

Javier Rojo, Enrique Moguel*, David Valencia, Javier Berrocal, Jose García-Alonso* and Juan M. Murillo

SPILab - Social and Pervasive Innovation Lab. Quercus Software Engineering Group. Escuela Politécnica. Avenida de la Universidad s/n. University of Extremadura. 10004 - Cáceres. Spain
* Corresponding Author: enrique@unex.es / jgaralo@unex.es

Abstract

Quantum computing holds great promise to solve problems where classical computers cannot reach. To the point where it already arouses the interest of both scientific and industrial communities. Thus, it is expected that hybrid systems will start to appear where quantum software interacts with classical systems. Such coexistence can be fostered by service computing. Unfortunately, how quantum code can be offered as a service still misses out on many of the potential benefits of service computing. This paper takes the traveling salesman problem and tackles the challenge of giving it an implementation in the form of a quantum microservice. Then it is used to detect which of the benefits of service computing are lost in the process. The conclusions help to measure the distance between the current state of technology and the state that would be desirable to have a real quantum service engineering.

1. Introduction

Quantum computing has been a relevant research field for more than 20 years, bringing together classical information theory, computer science, and quantum physics [1]. More recently, the development of quantum computers has brought us to the Noisy Intermediate-Scale Quantum (NISQ) era [2], where quantum computers with more than 50 qubits, although limited by the noise in quantum gates, are starting to perform tasks that may surpass the capabilities of classic computers.

Alongside this scientific development, quantum computing is also experiencing a significant commercial growth [3]. Several major computing corporations have built their own quantum computers and are offering them to users, mostly in a pay-per-use model. Engineers have designed and implemented dozens of quantum programming languages, simulators, and toolkits. All of this is paving the way for the development of quantum software and services.

Nevertheless, for the time being, classical and quantum services must not only coexist but interact with each other [4]. This coexistence has been called by some researchers hybrid

---

classical-quantum systems [5, 6]. A natural way to approach such collaborative coexistence is by following the principles of service engineering and service computing.

Already, companies and researchers are leaning towards the use of quantum infrastructure and quantum software as a service, as they are used to do with classical computing resources. Offerings like IBM Quantum Computing[1] or Amazon Braket[2] allow them to use the still very expensive to own and operate quantum computers with moderated costs. This model also fits very well with the needs of hybrid systems where both classical and quantum software will be executed on hardware on the cloud regardless of the type of computer needed.

Such deployment architectures are also perfectly aligned with the microservices architectural pattern [7]. Following this architecture, a complex system is conceived as a set of distributed microservices, where each microservice is a cohesive, independent process that interacts with the rest of the system through messages. Bringing microservices to hybrid systems, we have solutions where both classical and quantum services coexist to solve complex problems.

To achieve these hybrid microservices architectures, the first step is to convert a quantum piece of software into a microservice. Conceptually, there is no difference between a classical and a quantum microservice, an independent process that can interact with the rest of the system through messages. Furthermore, from a service engineering point of view the specific hardware in which a microservice is executed, classical or quantum, should be irrelevant. However, the current state of quantum services is very different from classical services and requires specific approaches to create working hybrid microservices architectures.

Running a quantum algorithm as a microservice is possible. The quantum algorithm can be wrapped by a classical service and integrated into the complex architecture. However, the current technological state of the available quantum platforms imposes some limitations. First, they make quantum services invocation and execution to be closely coupled with the quantum processor in which they will be executed. Also, the different quantum computers provide the results of the executed process in different ways thus making even stronger coupling. In addition, due to the existing noise in quantum computers, results are subject to errors and these errors are also usually dependent on each specific quantum computer and qbit topology. This increases, again, the coupling between service and hardware. Finally, due to the quantum system collapse, is not always possible to obtain intermediate verification of results which drastically reduces services orchestration possibilities.

For all these reasons, invoking a quantum microservice in an agnostic way is not possible and violates all the principles of software engineering. These limitations mean that most of the advantages of service-oriented computing are lost when involving quantum microservices. Especially, those related to the different software quality x-abilities like composability, maintainability, reusability, modularity, etc. To address this situation, techniques of classical service engineering should be brought to the domain of quantum service engineering [8].

Specifically, in this paper, we focus on the technical aspects needed to create a quantum microservice. To illustrate the current state of technology, we have developed a quantum microservice to solve the well-known Traveling Salesman Problem (TSP). We use the Amazon Braket platform to deploy it and to analyze its characteristics in the different quantum hardware

---

[1]https://www.ibm.com/quantum-computing/
[2]https://aws.amazon.com/braket/

supported by Amazon. Although a specific algorithm (TSP) and platform (Amazon Braket) are used, we believe that the problems and limitations found are directly transferable to other quantum algorithms and platforms. From the results obtained from executing this microservice, we discuss the current limitations of hybrid microservices architectures and detect which of the benefits of service computing are lost in the process.

In order to do that, the rest of the paper is organized as follows. Section 2 present the background of this work. Section 3 details the traveling salesman problem used as the case study for this work, paying special attention to the quantum implementations of this well-known problem. Section 4 presents how to offer a quantum algorithm as a microservice using the Amazon Braket platform. Section 5 lists the main results obtained by executing the quantum microservices in the different hardware and discusses the current limitations of quantum microservices. And finally, Section 6 presents the paper conclusion and future works.

2. Background

Microservices in particular, or Service-Oriented Architecture in general, is a software engineering approach focused on the use of services as the fundamental element to develop software solutions [9]. Although different definitions and proposals can be found in the literature, there are some aspects of microservices that are mostly agreed upon.

First, a microservice can be defined as a single-responsibility entity that encapsulates data and logic. They are exposed remotely and can be deployed, changed, substituted, and scaled independently of each other [9].

When developing microservices solutions, different computing paradigms and storage paradigms can be used [10]. Different programming languages are used, including a mix of functional and imperative ones, and databases, including relational and NoSQL ones, to provide solutions to complex problems.

There is no standard communication mechanism for microservices. Nevertheless, in practice, REST HTTP and asynchronous message queues are the most commonly used ways to expose microservices [11].

Similarly, although there is no constraint on where and how microservices should be deployed, in practice most solutions are developed with a strong orientation towards the cloud [12]. The elasticity and distribution provided by the cloud are features well aligned with the microservices approach.

Finally, although they are completely independent paradigms that are not exclusively related to microservices, continuous delivery and DevOps approaches are usually applied during the development of microservices systems [13].

Having all this into account, we can assume that microservices will be a good fit for a hybrid classical-quantum solution. In this regard, some of the most recent works on quantum software development are helping to align both worlds. Especially, from a cloud computing perspective.

Currently, most commercial quantum computers are accessible through the cloud, similar to the classical Infrastructure as a Service model. Some researchers have called this access Quantum Computing as a Service (QCaaS) [14]. Through QCaas developers can use some of the existing quantum computers to execute their own code. However, this access is still very

dependant of the specific hardware, requiring developers to have a deep understanding of it.

To address some of the QCaaS limitations and increase its abstraction level to create more complex quantum software, multiple research and commercial efforts are underway. In the academic world, quantum software engineering is starting to emerge, attracting the attention of researchers [15, 16]. This discipline seeks to bring the knowledge and expertise of classical software engineering to the domain of quantum software development. Specifically, some works are starting to pay attention to aspects of quantum development more closely related to microservices. For example, in [17] the authors propose the term Quantum application as a Service (QaaS) to narrow the gap between classical service engineering and quantum software. Also, [18] proposes an extension to TOSCA, a standard for software deployment on the cloud, to allow the deployment of quantum services.

From a different point of view, , quantum computers with practical utility and hybrid classical-quantum algorithms are emerging, such as Quantum Approximate Optimization Algorithm (QAOA) |[19, 20] or Variational-Quantum-Eigensolver (VQE) [21, 22].

In addition, companies are also trying to create more complex quantum solutions. Amazon, one of the global leaders in the cloud and computing services domains, has created the Amazon Braket platform that provides a development environment for quantum software engineers. At the moment of writing this article, Braket supports hardware from three different vendors (D-Wave, IonQ, and Rigetti). D-Wave machines fall in the category of adiabatic quantum computers, while IonQ and Rigetti machines can be classified as circuit-based quantum computing. Having these two different computational models available increases the tools at the disposal of quantum software developers, but also increase the complexity of programming quantum services. To use adiabatic-based machines developers must reformulate the problem they want to solve as a quantum annealing metaheuristic specification [23]. To use circuit-based machines developers should know the details of quantum gates and how to adapt the problem they want to solve to a quantum circuit [24]. This makes it more complicated for developers to create independent, maintainable, agnostic quantum microservices.

Other companies are creating similar platforms, like the above-mentioned IBM Quantum or like QPath[3]. However, as far as the authors know, there is no proposal that has addressed the problems and limitations of creating quantum microservices for the development of hybrid solutions.

3. Traveling Salesman Problem

To illustrate the technical limitations of current quantum platforms for the development of quantum microservices we have chosen a very well-known and studied problem. The traveling salesman problem is a famous example in the class of NP-Class problems [25]. It can be categorized as an optimization problem, in which the traveling salesman must visit all cities inside a route, minimizing the traveled distance. Thus, in the classical definition of the problem there exist cities, usually described as nodes, roads connecting those cities, that can be considered as links between these nodes each with a weight indicating the distance. The main drawback of these kinds of problems lies in the increasing of possible solutions with the increase of the

---

[3]https://www.quantumpath.es/

problem size, i.e. with 5 cities there exist 12 possible routes whereas for 25 cities the number of routes grows to 3.1×10^{23}.

Solving this problem by classical computing methods it is not always optimal. These methods have been developed for years as replacements to brute force solutions on these optimization problems, but still have certain limitations. In recent years, due to the expansion of quantum computing, researchers have begun to develop quantum algorithms that solve these problems: both for the perspective of adiabatic quantum computing [26] and for the perspective of gate-based quantum computing [27, 28].

3.1. Formulation of the TSP problem

To formulate the problem we have followed the previous works [29, 30]:

Definition 1. *Let $G = (N, A)$ be a directed graph, where $N = \{0, 1, ..., n\}$ is the finite set of nodes, also known as cities, and $A = N \times N$ is the set of roads or arcs connecting the cities. For every pair (u, v) of cities there exists a road in A. A tour is defined by the order in which the cities are visited.*

Definition 2. *Let city 0 denote the depot and assume that every tour begins and ends at the depot. Each of the remaining n cities appears exactly once in the tour. We denote a tour as an ordered list $P = (p_0, p_1, ..., p_n, p_{n+1})$, where p_i is the index of the city in the i-th position of the tour. According to our previous assumption $p_0 = p_{n+1} = 0$, which can also be indicated as $p_i(mod(n+1))$.*

Definition 3. *For every pair (u, v) of cities $u, v \in N$, there is a cost $c_{u,v}$, for traversing the road (u, v). This cost of traversing the road from u to v generally consists of the travel time from city u to city v.*

Based on the previous definitions, the objective function is to minimize the sum of the arc traversal costs along the tour, and can be summarized as:

$$min \sum_{i=1}^{n+1} c_{p-1,p} \qquad (1)$$

In eq. (1), it is assumed that $(p_0, p_1, ..., p_n, p_{n+1})$ is a feasible tour.

3.2. TSP on Quantum Computing Architectures

The classical formulation of the problem is proposed as an optimization, making it extremely suitable and straightforward for adiabatic quantum computers. More challenging is the proposal of solutions tailored for quantum computers based on quantum circuits and gates. In this subsection, a solution for the TSP on each type of quantum computers is described.

3.2.1. TSP on Adiabatic Quantum Computing

For this type of quantum computer, Amazon Braket provides a proposal of a solution based on quantum annealing. This proposal employs the Lagrange multipliers and Quadratic Unconstrained Binary Optimization (QUBO) problem matrix [31] where the graph of the problem is encoded such that the evolution of the system through quantum annealing offers the minimum energy cost, that is to say, the minimum travel cost.

To do this, the graph is internally converted to the form of an Ising model or Quadratic Unconstrained Binary Optimization Problem (QUBO), and later on, quantum annealing is applied.

As input for the algorithm, it is necessary to provide a matrix with the costs of traveling between cities. An example of this matrix is showed in Figure 1-(a). From it, a graph like the one on Figure 1-(b) can be generated.
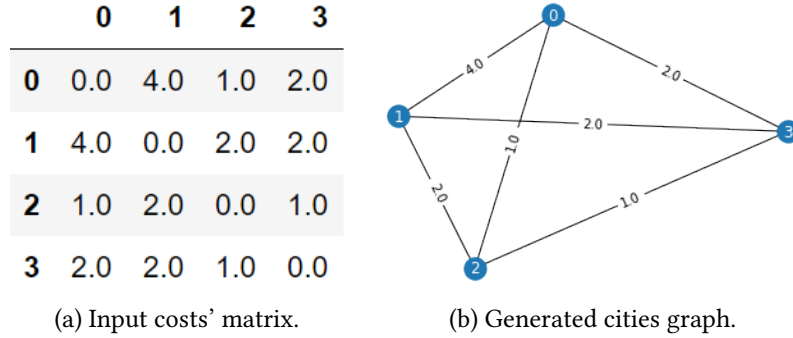

	0	1	2	3
0	0.0	4.0	1.0	2.0
1	4.0	0.0	2.0	2.0
2	1.0	2.0	0.0	1.0
3	2.0	2.0	1.0	0.0

(a) Input costs' matrix. (b) Generated cities graph.

Figure 1: Input for the annealing-based solution of the TSP.

As output, the Hamiltonian cycle with the lowest cost is returned [32] with 2 considerations: including revisiting the starting point or not revisiting it, i.e., in Figure 2 it is provided a graphic representation of the path with less cost, without returning to initial point.
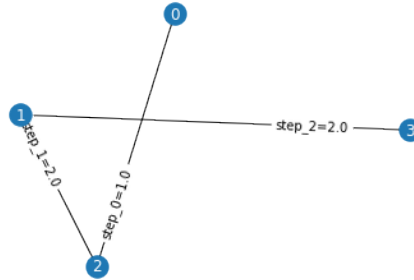

Figure 2: Output of the annealing-based solution of the TSP

3.2.2. TSP on Gate-based Quantum Computing

Solving the TSP in quantum machines programmed via quantum gate-based circuits is much more complicated. Optimization problems, such as TSP, can be addressed simply in adiabatic quantum computers by their nature, but to work on gate-based quantum machines a workaround is needed. This is related to the complexity of defining a circuit that allows finding the solution to the problem in any situation, obtaining in many cases circuits that only solve the problem for a specific case.

There are different approaches in the literature to solve the TSP in gate-based quantum computers, by means of the application of different quantum algorithms and solutions. In this

work, the circuit proposed by [28] will be used. It is based on the Quantum Phase Estimator (QPE) algorithm, which calculates a phase for each of the eigenstates considered. These eigenstates correspond to each of the possible Hamiltonian cycles solutions of the problem. Having obtained the phase of each eigenstate, they are later checked to select the lowest using other quantum algorithms such as the minimum finding [33]. Thus, the optimal path is the one for which the QPE obtains a minimum phase. This quantum algorithm has been implemented in Amazon Braket.

As opposed to the quantum annealing solution, the circuit obtained is not generic, one circuit must be generated for each eigenstate and an implemented circuit is associated with a concrete graph due to the nature of the controlled-U gates and the Unitary matrices obtained. The included code fragment is tailored for the graph shown in Figure 1. This way, subsequent experiments, and the obtained results will be easier to compare. Nonetheless, one must have in mind that if any of the elements of the graph change, the circuit should be changed as well.

After obtaining the result of the quantum part of the algorithm, classical computations must be applied to determine which eigenstate has as result the phase of smaller value and, knowing which is that eigenstate, return the associated Hamiltonian cycle.

Analyzing both solutions to the TSP, it is clear that the main complication lies in the case of quantum gate-based machines. Making generic circuits, which enable to take as input a series of parameters that condition them —in this case, a circuit for the QPE algorithm that works with any graph—, is something that has already been studied in proposals like [27, 34] that use Parameterized Quantum Circuits. Specifically, Matsuo et al. [27] propose a circuit to solve the TSP for any input graph in gate-based quantum machines, using the VQE algorithm. In this work solutions of this type have not been used since it has not been considered essential for the performed experiments. Being able to solve a given TSP is enough to build a microservice and analyze it.

4. Quantum microservices

From the two solutions to the TSP discussed in the previous section, a microservice with two endpoints has been implemented as shown in Figure 3. The implementation of this microservice and the notebooks with the implementations of the TSP in adiabatic and gate-based quantum computing are available in the following repository[4].

The two endpoints allow other microservices of the system to invoke the solution of the TSP problem, one using the gate-based solution and the other the adiabatic solution. Both endpoints are implemented in Python, using Flask[5]. This endpoint is in charge of deploying and executing the corresponding quantum algorithm.

These algorithms are executed on Amazon Braket and, therefore, different quantum computers can be chosen. In the case of the algorithm for adiabatic quantum computing, Braket supports the execution on the quantum computers *D-Wave 2000Q* and *D-Wave Advantage_system*. In the case of gate-based quantum computing, it can be executed on the simulators *LocalSimulator*, *TN1*, and *SV1* or on the quantum computers *IonQ*, *Rigetti Aspen-8*, and *Rigetti Aspen-9*. In any

---

[4]https://github.com/frojomar/ICWS2021-quantum-classical-microservices
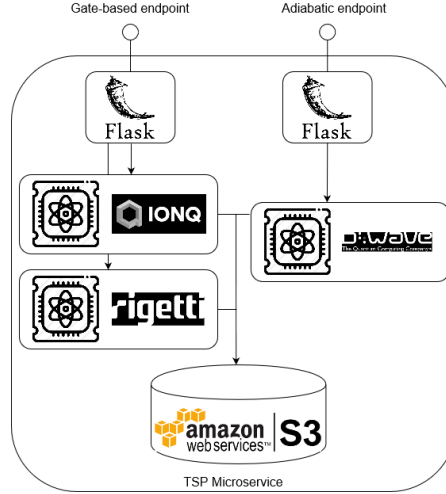[5]https://flask.palletsprojects.com/en/1.1.x/

Figure 3: Deployment architecture of the TSP hybrid microservice

case, the result of the execution of the quantum algorithms is always stored in an Amazon S3 storage (except in the case of *LocalSimulator*).

In order to choose over which hardware the microservice's quantum part is going to be executed, both endpoints employ a parameter. This parameter is codified as a *query param* called *device*. Taking into account the number of QPUs and simulators available, the query param *device* must take one of the following values:

- Adiabatic quantum computing endpoint: *dwave_dw2000*, and *dwave_advantage*.
- Gate-based quantum computing endpoint: *local*, *tn1*, *sv1*, *ionq*, *riggeti_aspen8*, and *riggeti_aspen9*.

Next, implementation details of both endpoints will be provided showing that, despite the fact that each one of them belongs to a different type of quantum computing, the way of enclosing the quantum code with a classical computing wrapper, that allows its execution as a microservice, does not differ.

4.1. Adiabatic quantum computing endpoint

Figure 4 shows the code for the endpoint of the adiabatic solution.

As input, the endpoint needs a .txt file with the weights of the matrix, in addition to the query param *device*. As a result, a JSON object is returned including the best route found for the TSP problem and the distance to cover the route.

4.2. Gate-based quantum computing endpoint

As in the previous endpoint, Figure 5 shows the code for the endpoint of the circuit-based solution.

```python
@app.route('/execute/adiabatic', methods=["get"])
def execute_adiabatic_quantum_tsp():
    machine = request.args.get('device')

    if machine not in adiabatic_machines_arn:
        return "Not machine", 400

    if 'file' not in request.files:
        return "Not file", 400
    file = request.files['file']
    if file.filename == '':
        return "Not file", 400

    filename = "graph.txt"
    file.save(filename)

    data, G, weights = def_graph(filename)

    best_route, best_distance = TSP(data, G, weights,
                                    s3_folder,
                                    adiabatic_machines_arn[machine])

    response = {
        "best_route":best_route,
        "best_distance":best_distance
    }
    return jsonify(response)
```

Figure 4: Adiabatic quantum computing endpoint code

```python
@app.route('/execute/gate', methods=["get"])
def execute_gate_based_quantum_tsp():

    machine = request.args.get('device')

    if machine not in gate_machines_arn:
        return "Not machine", 400

    eigenstates = ["11000110", "10001101", "11100001"]
    results = {}
    for e in eigenstates:
        counts = QPE(e, s3_folder, gate_machines_arn[machine])
        measure = list(counts.keys())[0]
        eigenstate = measure[n_ancilla:]
        phase = measure[:n_ancilla]
        results[eigenstate] = phase

    hamiltonian_cycle = get_minor_route_by_phase(results)

    response = {
        "best_route":hamiltonian_cycle
    }
    return jsonify(response)
```

Figure 5: Gate-based quantum computing endpoint code.

Depending on whether the TSP is to be executed on a real quantum computer or on a simulator, the way in which the quantum code is executed changes. Therefore the selected machine is given as a parameter to the method that makes the quantum call. When the endpoint is run outside of a local simulator the result is stored in s3 storage, from where it is retrieved. In particular, if it is run on a quantum computer, the results are always stored in an s3 and, in addition, they take some undefined time to be available.

When running the code on a quantum computer, a task with an identifier is created. With this identifier, developers can check the status of the task, which will change from CREATED,

QUEUED, and RUNNING until it reaches COMPLETED; or CANCELLED or FAILED if something goes wrong. At this point, the result can be recovered. In any case, it is necessary to define a poll timeout to prevent the execution from being blocked.

All this is done transparently to the system since, when the microservice is invoked, it is checked with which device the code is to be executed and, if it is a real quantum computer, the wait is done internally in the call to the endpoint.

In order to execute this endpoint, it is necessary to send the query param *device* with the device where the quantum algorithm is to be executed. In this case, the quantum circuit is not parameterized, so it does not allow the execution on different networks. That same graph will be always used as mentioned above. As output, this endpoint returns the optimal path but not the path cost, as the algorithm does not know the weight of each path.

5. Quantum-Classical Hybrid Microservices System Trial Evaluation

After developing the described microservice, different metrics were used to evaluate its performance and the limitations of including a quantum microservice in a hybrid system.

To proceed with the evaluation, several HTTP requests were made from the API client tool Postman[6], which allows performing petitions to REST APIs and take metrics such as response time, response size, etc.

The developed microservice was locally deployed during the evaluation. More specifically, it was deployed on a laptop running Windows 10 with 16 GB of RAM and an Intel Core i7-8550U processor at 1.8 GHz base frequency and 4.0 GHz turbo frequency, equipped with NVMe SSD technology for storage.

Taking advantage of the fact that the adiabatic solution implemented for the TSP is generic and the endpoint allows giving a graph as input, the evaluation for both adiabatic and gate-based implementations has been carried out using the same graphs as input. Thus, the comparison between the results obtained with both computation models is comparable.

Table 1 summarizes the launched executions and the corresponding results obtained. The first 2 lines correspond to the Adiabatic Quantum Endpoint and the rest of the table is related to the Gate-based Quantum Endpoint.

Number of qubits. One of the main limitations of current quantum computers lies in the number of qubits available, especially in the case of gate-based systems, and this directly affects the ability to run the microservice or limits its execution. Table 2 shows the results from considering the TSP described in Figure 1. As one can see, in the case of Gate-based circuits the number of qubits amounts to 14 (8 for eigenstates + 6 for phase). In the case of the quantum annealing solution, the number of qubits necessary is unknown, since the Braket provided implementation was used. Nevertheless, both D-wave machines available at Braket had enough qbits to run the algorithm. In any case, even for such a simple problem (3 possible routes considering the links between 2 nodes as symmetric non-directed), it exceeds the number of qubits available (11 qubits) to be executed on the IonQ hardware and it is not possible to execute on this architecture. In the case of the Rigetti hardware, it provides enough qubits.

---

[6]https://www.postman.com/product/api-client/

Architecture	# of shots	Result obtained
DWAVE 2000Q6	$10^2, 10^3, 10^4$	[0,3,1,2] (Consistent)
DWAVE ADVANTAGE	$10^2, 10^3, 10^4$	[0,3,1,2] (Consistent)
LocalSimulator	$10^3, 10^4, , 10^5$	[0,3,1,2], [0,1,2,3]
TN1	—	Error
SV1	$10^3, 10^4$	[0,1,2,3] (Inconsistent)
IonQ	—	Error
Aspen 8	—	Error
Aspen 9	10^3	Error

Table 1
Executions on each endpoint and shots conducted.

Version	# of Qubits	# of Classical bits
Gate-based TSP	14 (eigenstates+ phase)	6 (collapsing phase)
Dwave's solution	Unknown	Unknown

Table 2
Executions conducted and number of qubits and classical bits

In the case of quantum services, this not only shows the limited power of the current hardware but also the need that quantum service engineering will have for mechanisms to determine the number of qubits the execution of service will need. Due to the nature of quantum algorithms for the different architectures, there is no trivial way to obtain this number. This will be a key question in developing quantum services execution planners with implications in several other aspects of the service like if the cost of only the hardware with more qubits can be used or response time if the waiting time for that hardware is longer.

Number of shots. Due to the problems that arise due to the characteristics of actual quantum computers, mainly noise in the qubits state, the experiments must be conducted several times or "shots" to be statistically consistent.

For a real quantum service technology, the responsibility of performing the different executions to get a consistent result cannot be delegated to the client nor the customer who only wants to use technology to get a correct result, at least within a given margin of error, and with an economic cost known in advance. How the number of shots required is estimated will have a direct impact on the cost of the service executions. This reveals some issues, related to service quality and costs, that still have to be addressed by quantum services engineering.

Precision of results. Table 1 shows the discrepancies in the results achieved. In the first rows, the results obtained by DWave's machines are shown. Both show consistent results given the number of shots considered. For the rest of the platforms, different problems have arisen. First, in some of the architectures, it has been impossible to execute the code, more specifically it happened in TN1, IonQ, and Rigetti 8. On IonQ the number of qubits available was insufficient to run the service. On Apen 8, the service was unavailable at the time of running the experiments. Finally, in Aspen 9, although the code was sent for execution, it was in the state QUEUED for more than 3 hours, not having executed any of the 1000 shots. In contrast, in the case of SV1, the code was sent and executed. However, the results obtained were different in different

executions. Most of the time the result was [0,1,2,3] which does not correspond to the optimal solution. Similar results were observed when working with the simulated architecture, where the number of shots must be higher than 100 to obtain the correct solution with acceptable statistical certainty.

Again, in a real scenario, the responsibility of determining the number of shots and the precision of the results cannot be delegated to the client service nor the customer. The customer pays for a service that is expected to provide correct results, within an agreed level. This points out the need for some kind of logic in service execution planners to determine the number of shots needed to provide a correct solution. It should also be noted that predictions can affect the planning, availability, and accuracy of the platforms' results and all of this will impact the service qualities that will have been previously negotiated with the customers. These are therefore issues that, while affecting the technical aspects of future quantum services platforms, will also affect their financial profitability.

Response times. Other evaluated parameter was response time. The measure corresponds to the time elapsed between sending the request and receiving the result. This time has been measured for all the machines where the code has been correctly executed. Specifically, the SV1 and LocalSimulator gate-based simulators, and the D-wave 2000_Q6 and D-wave Advantage adiabatic quantum machines. For the first ones, the difference is significant. In the LocalSimulator, the execution took about 3 seconds with 1000 and 10000 shots, and about 7 seconds with 100000 shots. However, in SV1 it took an average of 27 seconds, with a margin of up to 10 seconds between the fastest and slowest runs. In the case of adiabatic machines, the result is similar for both. In the case of D-wave 2000_Q6, the runs exceed 20 seconds, and in the case of D-wave Advantage 25 seconds.

From this, it can be concluded that the highest cost in terms of time is incurred when sending the quantum code to execution. Possibly, due to the waiting times in the queue. Nevertheless, these results give the user a feeling of unreliability when using the platforms which, in real service engineering, must be avoided. Dealing with this will again require quantum service platform planners to count on reliable resources and estimates. This highlights how far current quantum service platforms are from reaching to be acceptable to potential customers of a quantum services platform.

Economic cost. Lastly, the economic cost of invoking each solution has been considered. At the moment, Amazon Braket establishes a fixed price per quantum task executed, which is the same for all the supported hardware. Moreover, an additional cost is paid for each shot and this cost is different for the different hardware. These costs, while predictable if the hardware to be used and the number of shots to be executed are known, are far from what is needed to agnostically implement microservices on hybrid quantum-classic systems. Especially due to the uncertainty that arises from the unavailability of services, response time, uptime, state of the quantum system, and so on, parameters are extremely important to be able to assure the quality and SLAs inherent in services.

Furthermore, from the evaluation performed, some other more abstract questions arise as well. First and foremost, there is a need for abstractions to define quantum problems in a more general way. This abstraction could be used as a starting point that can be specialized in terms of quantum annealing, gate-based circuits, or whatever future technology or new programming paradigm appears for quantum computing. An initial solution, for the specific case presented in

this paper, could be to develop a single generic endpoint in charge of unifying the adiabatic and gate-based solutions. Such an endpoint will have the responsibility of adopting an abstract representation of the TSP problem to the needs of the specific quantum hardware in which it will be executed. Even then, the solution will still depend on the service platform, Braket in this case. This remarks one unresolved question in quantum service engineering. When a service is invoked, the invoker only cares for the response. The service platform should address the execution of different architectures and the problem formulation for each of them if there is a benefit in doing so. Delegating these responsibilities to the client makes it more tightly coupled with the microservice and reduces the benefits provided by service engineering.

To summarizing, given the above-mentioned problems, current quantum services platforms pose the following inconveniences for the development of quantum microservices. First, services are tightly coupled with the quantum code to be executed. Moreover, services are also tightly coupled with the hardware in which the quantum code will be executed. Additionally, platforms do not allow a service implementation to be transparently replaced by another, as can happen in traditional services as long as the API is maintained. Also, quantum platforms are not able to decide, on execution time, where and how a service will be executed to optimize answering petitions based on performance aspects of the different supported hardware. Finally, all the experiments developed in this paper involve only a single service which is completely unreal. The most simple example of a real microservices-based system would involve several ones. However, there is also an absolute lack of mechanisms for quantum services orchestration. All of these limitations have a significant impact on some of the most relevant aspects of quality services, like composability, maintainability, reusability, or modularity of quantum services limiting the current potential of these services. These limitations affect not only researchers or developers but also the platforms that provide access to quantum hardware. The commercial success of cloud computing and services is supported by the elasticity provided to developers and the optimization of hardware usage provided to the hardware owners. Similar levels of flexibility and optimization should be possible in the quantum domain but additional research efforts are needed in quantum service engineering.

6. Conclusion and future works

In this paper, we have presented an implementation of a quantum microservice and the problem that arises from trying to integrate it in a hybrid microservices architecture. We have used Amazon Braket to test the implemented microservice on quantum hardware from three different vendors and to detect current limitations in the domain of hybrid classical-quantum microservices.

The performed experiments have allows us to clearly show the limitations of the current quantum computers platform for the development and exploitation of quantum microservices. Intense research efforts are still needed to bring the benefits of service-oriented computing and microservices to the quantum computing domain.

Since quantum software engineering is still a very young discipline, most of the areas that compose it are just starting to attract the interest of researchers, including hybrid microservices architectures. However, the change in the computing paradigm that implies quantum computing

means that we cannot directly translate the techniques and tools of classical microservices and expect them to work flawlessly in the new environment. Putting a quantum algorithm inside a microservice is not enough to create a quantum microservices, there needs to be an effort to generate new knowledge, techniques, methodologies,... that helps bridge the gap between classical microservices and the advantages of quantum computers.

Acknowledgments

References

[1] A. Steane, Quantum computing, Reports on Progress in Physics 61 (1998) 117.

[2] J. Preskill, Quantum computing in the nisq era and beyond, Quantum 2 (2018) 79.

[3] E. R. MacQuarrie, C. Simon, S. Simmons, E. Maine, The emerging commercial landscape of quantum computing, Nature Reviews Physics 2 (2020) 596–598.

[4] B. Sodhi, Quality attributes on quantum computing platforms, arXiv preprint arXiv:1803.07407 (2018).

[5] A. McCaskey, E. Dumitrescu, D. Liakh, T. Humble, Hybrid programming for near-term quantum computing systems, in: 2018 IEEE International Conference on Rebooting Computing (ICRC), IEEE, 2018, pp. 1–12.

[6] A. J. McCaskey, D. I. Lyakh, E. F. Dumitrescu, S. S. Powers, T. S. Humble, Xacc: a system-level software infrastructure for heterogeneous quantum–classical computing, Quantum Science and Technology 5 (2020) 024002.

[7] N. Dragoni, S. Giallorenzo, A. L. Lafuente, M. Mazzara, F. Montesi, R. Mustafin, L. Safina, Microservices: yesterday, today, and tomorrow, Present and ulterior software engineering (2017) 195–216.

[8] M. Piattini, G. Peterssen, R. Pérez-Castillo, J. L. Hevia, M. A. Serrano, G. Hernández, I. G. R. de Guzmán, C. A. Paradela, M. Polo, E. Murina, L. Jiménez, J. C. Marqueño, R. Gallego, J. Tura, F. Phillipson, J. M. Murillo, A. Niño, M. Rodríguez, The talavera manifesto for quantum software engineering and programming, in: Short Papers Proceedings of the 1st International Workshop on the QuANtum SoftWare Engineering & pRogramming, Talavera de la Reina, Spain, February 11-12, 2020, volume 2561 of *CEUR Workshop Proceedings*, 2020, pp. 1–5.

[9] O. Zimmermann, Microservices tenets, Computer Science-Research and Development 32 (2017) 301–310.

[10] J. Bogner, A. Zimmermann, Towards integrating microservices with adaptable enterprise architecture, in: 2016 IEEE 20th International Enterprise Distributed Object Computing Workshop (EDOCW), IEEE, 2016, pp. 1–6.

[11] J. Bogner, J. Fritzsch, S. Wagner, A. Zimmermann, Microservices in industry: insights into technologies, characteristics, and software quality, in: 2019 IEEE International Conference on Software Architecture Companion (ICSA-C), IEEE, 2019, pp. 187–195.

[12] A. Sill, The design and architecture of microservices, IEEE Cloud Computing 3 (2016) 76–80.

[13] L. Chen, Microservices: architecting for continuous delivery and devops, in: 2018 IEEE International conference on software architecture (ICSA), IEEE, 2018, pp. 39–397.

[14] M. Rahaman, M. M. Islam, A review on progress and problems of quantum computing as a service (qcaas) in the perspective of cloud computing, Global Journal of Computer Science and Technology (2015).

[15] J. Zhao, Quantum software engineering: Landscapes and horizons, CoRR abs/2007.07047 (2020). URL: https://arxiv.org/abs/2007.07047. arXiv:2007.07047.

[16] M. Piattini, G. Peterssen, R. Pérez-Castillo, Quantum computing: A new software engineering golden age, ACM SIGSOFT Softw. Eng. Notes 45 (2020) 12–14. URL: https://doi.org/10.1145/3402127.3402131. doi:10.1145/3402127.3402131.

[17] J. Barzen, F. Leymann, M. Falkenthal, D. Vietz, B. Weder, K. Wild, Relevance of near-term quantum computing in the cloud: A humanities perspective, in: D. Ferguson, C. Pahl, M. Helfert (Eds.), Cloud Computing and Services Science - 10th International Conference, CLOSER 2020, Prague, Czech Republic, May 7-9, 2020, Revised Selected Papers, volume 1399 of *Communications in Computer and Information Science*, Springer, 2020, pp. 25–58. doi:10.1007/978-3-030-72369-9_2.

[18] K. Wild, U. Breitenbücher, L. Harzenetter, F. Leymann, D. Vietz, M. Zimmermann, TOSCA4QC: two modeling styles for TOSCA to automate the deployment and orchestration of quantum applications, in: 24th IEEE International Enterprise Distributed Object Computing Conference, EDOC 2020, Eindhoven, The Netherlands, October 5-8, 2020, IEEE, 2020, pp. 125–134. doi:10.1109/EDOC49727.2020.00024.

[19] G. G. Guerreschi, A. Y. Matsuura, QAOA for Max-Cut requires hundreds of qubits for quantum speed-up, Scientific Reports 2019 9:1 9 (2019) 1–7. URL: https://www.nature.com/articles/s41598-019-43176-9. doi:10.1038/s41598-019-43176-9.

[20] M. Streif, M. Leib, Comparison of QAOA with Quantum and Simulated Annealing (2019). URL: https://arxiv.org/abs/1901.01903v1. arXiv:1901.01903.

[21] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, J. L. O'Brien, A variational eigenvalue solver on a quantum processor, Nature Communications 5 (2013). URL: https://arxiv.org/abs/1304.3061v1. doi:10.1038/ncomms5213. arXiv:1304.3061.

[22] J. R. McClean, J. Romero, R. Babbush, A. Aspuru-Guzik, The theory of variational hybrid quantum-classical algorithms, New Journal of Physics 18 (2015). URL: https://arxiv.org/abs/1509.04279v1. doi:10.1088/1367-2630/18/2/023023. arXiv:1509.04279.

[23] S. Boixo, T. Albash, F. M. Spedalieri, N. Chancellor, D. A. Lidar, Experimental signature of programmable quantum annealing, Nature communications 4 (2013) 1–8.

[24] R. Wille, R. Van Meter, Y. Naveh, Ibm's qiskit tool chain: Working with and developing for real quantum computers, in: 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE), IEEE, 2019, pp. 1234–1240.

[25] R. H. Warren, Adapting the traveling salesman problem to an adiabatic quantum computer,

Quantum information processing 12 (2013) 1781–1785.

[26] R. H. Warren, Solving the traveling salesman problem on a quantum annealer, SN Applied Sciences 2 (2020) 1–5. URL: https://doi.org/10.1007/s42452-019-1829-x. doi:10.1007/s42452-019-1829-x.

[27] A. Matsuo, Y. Suzuki, S. Yamashita, Problem-specific Parameterized Quantum Circuits of the VQE Algorithm for Optimization Problems, arXiv (2020). URL: http://arxiv.org/abs/2006.05643. arXiv:2006.05643.

[28] K. Srinivasan, S. Satyajit, B. K. Behera, P. K. Panigrahi, Efficient quantum algorithm for solving travelling salesman problem: An IBM quantum experience, arXiv (2018). URL: http://arxiv.org/abs/1805.10928. arXiv:1805.10928.

[29] J. W. Ohlmann, B. W. Thomas, A compressed-annealing heuristic for the traveling salesman problem with time windows, INFORMS Journal on Computing 19 (2007) 80–90.

[30] C. Papalitsas, T. Andronikos, K. Giannakis, G. Theocharopoulou, S. Fanarioti, A qubo model for the traveling salesman problem with time windows, Algorithms 12 (2019) 224.

[31] P. Date, R. Patton, C. Schuman, T. Potok, Efficiently embedding QUBO problems on adiabatic quantum computers, Quantum Information Processing 18 (2019) 117. URL: https://doi.org/10.1007/s11128-019-2236-3. doi:10.1007/s11128-019-2236-3.

[32] A. Mahasinghe, R. Hua, M. J. Dinneen, R. Goyal, Solving the Hamiltonian Cycle Problem using a Quantum Computer, in: ACM International Conference Proceeding Series, Association for Computing Machinery, New York, NY, USA, 2019, pp. 1–9. URL: https://dl.acm.org/doi/10.1145/3290688.3290703. doi:10.1145/3290688.3290703.

[33] C. Durr, P. Hoyer, A Quantum Algorithm for Finding the Minimum (1996). URL: http://arxiv.org/abs/quant-ph/9607014. arXiv:9607014.

[34] A. P. Adelomou, E. G. Ribe, X. V. Cardona, Using the Parameterized Quantum Circuit combined with Variational-Quantum-Eigensolver (VQE) to create an Intelligent social workers' schedule problem solver, arXiv (2020). URL: http://arxiv.org/abs/2010.05863. arXiv:2010.05863.