Institute of Architecture of Application Systems

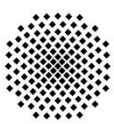
Quokka: A Service Ecosystem for Workflow-Based Execution of Variational Quantum Algorithms

Martin Beisel, Johanna Barzen, Simon Garhofer, Frank Leymann, Felix Truger, Benjamin Weder, Vladimir Yussupov

Institute of Architecture of Application Systems,
University of Stuttgart, Germany
{beisel, barzen, leymann, truger, weder, yussupov}@iaas.uni-stuttgart.de

BibTeX:

```
@inproceedings{Beisel2023_Quokka,
  author    = {Beisel, Martin and Barzen, Johanna and Garhofer, Simon and
               Leymann, Frank and Truger, Felix and Weder, Benjamin and
               Yussupov, Vladimir},
  title     = {Quokka: A Service Ecosystem for Workflow-Based Execution of
               Variational Quantum Algorithms}
  booktitle = {Service-Oriented Computing -- ICSOC 2022 Workshops},
  year      = {2023},
  month     = mar,
  pages     = {369--373},
  doi       = {https://doi.org/10.1007/978-3-031-26507-5_35},
  series    = {Lecture Notes in Computer Science (LNCS)},
  volume    = {13821},
  publisher = {Springer International Publishing}
}
```

University of Stuttgart
Germany

Quokka: A Service Ecosystem for Workflow-Based Execution of Variational Quantum Algorithms

Martin Beisel[1], Johanna Barzen[1], Simon Garhofer[2], Frank Leymann[1], Felix Truger[1], Benjamin Weder[1], and Vladimir Yussupov[1]

[1] University of Stuttgart, Universitätsstr. 38, 70569 Stuttgart, Germany
{lastname}@iaas.uni-stuttgart.de
[2] University of Tübingen, Sand 13, 72076 Tübingen, Germany
simon.garhofer@uni-tuebingen.de

Abstract. Hybrid quantum-classical applications are often implemented as monolithic applications that comprise various tightly-coupled classical and quantum tasks. However, the lifecycle of such applications can benefit from using service-oriented architectures, as they simplify scalable and resilient deployments and improve development processes by decoupling complex processes into more comprehensible work packages. In this demonstration, we (i) introduce Quokka, a service ecosystem that facilitates workflow-based development and execution of quantum applications by providing dedicated services for implementing each task in variational quantum algorithms. Further, (ii) we show how it can be used to orchestrate an example quantum application using workflows.

Keywords: Quantum Computing · Microservices · Workflows · SoC.

1 Motivation: Quantum Applications as Workflows

Quantum computing enables solving various problems with improved precision and in a more time- and energy-efficient manner by leveraging quantum mechanical phenomena, such as superposition and entanglement. However, quantum algorithms depend on multiple pre- and post-processing tasks that often need to be executed on classical hardware, e.g., data preparation, result analysis, and parameter optimization. As currently available *Noisy Intermediate-Scale Quantum (NISQ)* devices are error-prone, the majority of today's quantum algorithms are designed as so-called *Variational Quantum Algorithms (VQAs)* [2]. VQAs alternate between executing parameterized quantum circuits on a quantum device and classically optimizing the quantum circuit parameters by evaluating the quality of execution results. Moreover, quantum devices are not suitable for many traditional tasks, such as data persistence or visualization, which makes them rather special co-processors that complement classical computers. Quantum applications are, hence, inherently hybrid and must be designed with classical and quantum perspectives as well as their integration in mind [4].

While software engineering research is well-established for classical software, the topic of quantum software engineering is still in its early stage, with several works investigating the applicability of classical software engineering paradigms to quantum software [4]. In particular, workflow-based execution of quantum applications has multiple advantages brought by workflow technology, such as robustness, scalability, and reusability of created models [3]. To facilitate the modeling of quantum workflows, a quantum-specific BPMN extension has been developed, introducing a set of custom-tailored quantum-related tasks. Figure 1 shows an example workflow model for solving the *Travelling Salesman Problem (TSP)* using the *Quantum Approximate Optimization Algorithm (QAOA)*. As a prerequisite, users need to specify the TSP problem instance as an adjacency matrix and decide on hyperparameter for the workflow execution, which are passed to the workflow engine on workflow instantiation. After instantiation, the circuit for the given TSP problem instance is *(i) generated* and *(ii) executed*. Subsequently, occurred readout errors need to be *(iii) mitigated* and the execution result's quality is *(iv) evaluated*. Finally, the circuit parameters are *(v) optimized until convergence*. The result of the algorithm's last iteration is the final solution, which could be analyzed, e.g., by a following user task.

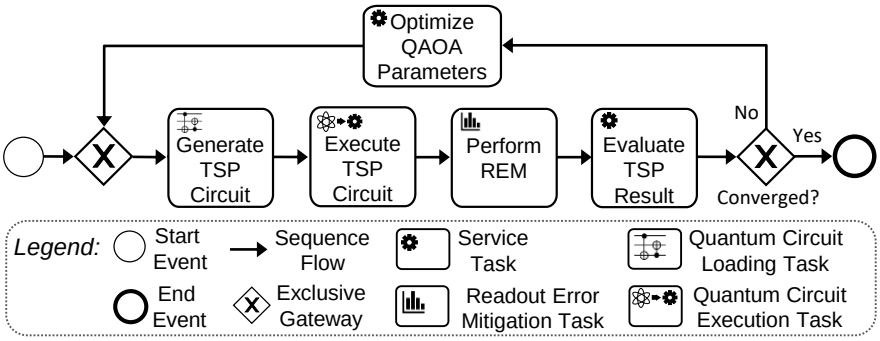


Fig. 1. Overview of a typical quantum workflow implementing a VQA.

The currently predominant way to implement such integrations of classical and quantum tasks is to use monoliths instead of workflows. However, hybrid quantum-classical applications can benefit from architectural styles such as microservice-based architectures that promote distribution, loose coupling, and better modularization. Employing these concepts leads to various advantages, such as reusability, maintainability, robustness, or scalability that improve the development and operations process. In particular, the reuse of existing and well-tested components leads to a cheaper, less repetitive and less error-prone development process with increased time-to-market. To tackle this problem, we (i) introduce QUOKKA, a service ecosystem that simplifies executing different tasks in VQAs, such as generation and execution of quantum circuits as well as readout error mitigation, and (ii) demonstrate how QUOKKA facilitates workflow-based execution of a VQA based on the scenario described in Figure 1.

2 The Quokka Ecosystem

The workflow model shown in Figure 1 describes the required steps to determine a suitable route for a given TSP problem instance. However, it does not provide an implementation of these steps, leaving the complex and time-consuming task of implementing them to the developer. By providing a set of microservices for each specific task type, the Quokka service ecosystem simplifies implementing these tasks. Quokka's system architecture is depicted in Figure 2, comprising the Quokka Gateway and five quantum task-specific microservices. All Quokka microservices are implemented in Python, enabling easy integration of the currently predominant Python-based quantum SDKs, including Qiskit and Braket. Furthermore, they follow a modular design, enabling developers to easily extend them and integrate code for specific use cases into the microservices.

Quokka Gateway: This Java-based API Gateway facilitates the request routing for clients, by uniting all Quokka microservices in a single REST API. It is built upon the *Spring Cloud Gateway* and intercepts and forwards incoming client requests to the respective microservices. Further, features brought by the Spring Cloud Gateway, e.g., monitoring or security, can be taken advantage of.

Circuit Generation Service: Composing quantum circuits is a complex task that is typically performed using quantum SDKs, which provide functions to generate complete quantum circuits but also allow the creation of scripts to assemble custom circuits. The circuit generation service provides on-demand circuit implementations for a set of quantum algorithms that users can extend by integrating their custom circuit generation scripts. API requests must include the required parameters for the generation process of the respective quantum algorithm and receive a circuit in the de facto standard language *OpenQASM*.

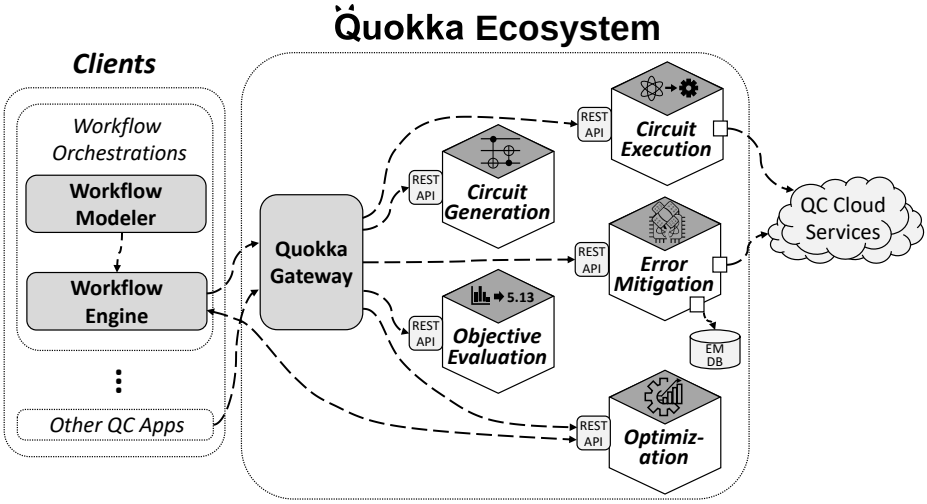


Fig. 2. Overview of the overall system architecture.

Circuit Execution Service: This service enables the execution of quantum circuits on various cloud-based quantum devices and simulators. In addition to error-free simulators, the service offers access to noisy simulators capable of simulating device errors, such that an algorithm's performance can be simulated without the common, long queuing times of current quantum devices.

Error Mitigation Service: To improve the quality of noisy execution results, various error mitigation methods, e.g., matrix inversion-based readout error mitigation, can be used via the error mitigation service [1]. These methods typically require up-to-date information about the used quantum device's error rates, which is obtained by executing additional circuits on the device. Since VQAs repeatedly execute similar circuits on the same device, error rates can be saved and reused, significantly improving the efficiency of the error mitigation process.

Objective Evaluation Service: This service enables a quality assessment of the execution result, which, for example, can be used for optimization or benchmarking. To assess an execution result's quality, first, a problem-specific cost function determines the cost of each measured bit string, e.g., traveled distances for a TSP. Subsequently, an objective function is used to compute a single value on the basis of the measurements' frequencies and costs, e.g., the expectation value, describing the overall quality of the result. Furthermore, a graphical representation of the result is provided, e.g., a graph highlighting the best route.

Optimization Service: This service provides a selection of optimizer implementations available for Python, including SciPy and Qiskit Terra optimizers. Clients initialize an optimization process by sending a request selecting their desired optimizer and initial parameters. Once an optimization process is initialized, messaging is used to exchange objective values and optimization parameters between the optimization service and the client, e.g., a workflow engine.

The QUOKKA source code repository, documentation, and a comprehensive tutorial can be found on GitHub: `https://github.com/UST-QuAntiL/Quokka`. The demonstration video is available on YouTube: `https://youtu.be/VQUz9Sj1r4M`.

Acknowledgments. This work was funded by the BMWK projects *PlanQK* (01MK20005N), *EniQmA* (01MQ22007B), and *SeQuenC* (01MQ22009B), and by the project *SEQUOIA* funded by the Baden-Wuerttemberg Ministry of Economic Affairs, Labour and Tourism.

References

1. Beisel, M., et al.: Configurable Readout Error Mitigation in Quantum Workflows. Electronics **11**(19) (2022)
2. Cerezo, M., et al.: Variational Quantum Algorithms. Nature Reviews Physics **3**(9), 1–20 (2021)
3. Weder, B., et al.: Integrating Quantum Computing into Workflow Modeling and Execution. In: Proceedings of the 13th IEEE/ACM International Conference on Utility and Cloud Computing (UCC 2020). pp. 279–291. IEEE (2020)
4. Weder, B., et al.: Quantum Software Development Lifecycle. In: Quantum Software Engineering. pp. 61–83. Springer International Publishing (2022)