

Hardware Agnostic Backend Adaptation through Quantum Circuit Evaluation

Abhishek Sharma^{1,*}, Kailash S^{2,**}, and Ethirajan D^{3,***}

¹Centre for Development of Advanced Computing (C-DAC), Chennai

²Centre for Development of Advanced Computing (C-DAC), Chennai

³Centre for Development of Advanced Computing (C-DAC), Chennai

Abstract. Quantum development platforms usually provide a choice of numerous simulators and quantum processing units (QPUs) for users to choose from and users are still forced to choose and configure a backend manually. As backend collections increase in size, the manual process consists of speculation or manual exploration with limitations in terms of performance and scalability. The approach specified in this paper, although applicable for major platforms, focuses on Qniverse, an unified quantum computing platform that has an integrated working environment for quantum circuit design and execution but still back ends are to be selected manually. The limitations of backend selection through manual conclusion are that novice users are affected and there are complexities in executing the operations. This paper introduces a hardware agnostic backend adaptation that automates the backend selection using circuit analyses and multi criteria backend ranking. The proposed model has three major modules. First is the Circuit Analysis Engine that identifies quantum circuit structure in terms of number of qubits, circuit depth and component gates. Second is a Backend Capability Mapping module that stores each backend capability details in structured forms. Third is an Intelligent Backend Recommendation Algorithm that specifically selects and weighs backend candidates according to latency time, accuracy level, cost constraints and resource allocation. All three unite to form an integrated quantum middleware adoption.

1 Introduction

With the advancement of quantum computing platforms, the user community is positioned in an environment with multiple backend involving the integration of perfect simulator systems, noisy simulator systems and heterogeneous quantum hardware platforms. In this scenario, apart from performance, accuracy and cost of the quantum workload depend upon the type of algorithms involved, also dependent upon additional parameters including qubits, connectivity topology, noise models, native gates and queue latency related to the hardware properties of the backend platforms. The comprehension of these aspects and subsequent optimizations involve hardware expertise, which most users, especially students, researchers from other fields and application developers lack.

*e-mail: sharma.abhishek@cdac.in

**e-mail: kailashs@cdac.in

***e-mail: ethirajand@cdac.in

The existing quantum environments, such as IBM Quantum, Qiskit run times, as well as cloud based quantum services, commonly provide backend choice capabilities based upon basic criteria like the number of qubits, availability or 'least busy' rules [1,3]. This creates a situation where users have to verify circuits over multiple backend to identify appropriate acceptable execution capabilities. This provides a platform for designing circuits, providing best suitable simulators and devices. However, backend choice is a manual or minimally automated process. This manual requirement creates three common problems. First, there is execution uncertainty, as users are not much aware about the backend providing the best execution or fidelity without manual verification. Second, scalability creates a problem as the choice of backend keeps growing. This technique will become inefficient as there would be a high chance of manual errors. Third, the learning as well as the barrier to entry will remain high for users who lack expertise on quantum hardware. To solve the aforementioned problems, a hardware agnostic adaptation is proposed that automatically analyze a submitted quantum circuit, comparing the capabilities for backend as per specific requirements. This proposed work will specifically focus on Hardware Agnostic Adaptation providing a basic layer for the Quantum circuit hardware improvement roadmap.

2 Literature Review

Hardware agnostic adaptation has been elaborated at great length in both high performance computing and cloud computing, introducing additional abstraction between applications and underlying hardware to enable portability and scalability. Analogous ideas are slowly making their way into the quantum domain as quantum classical hybrids are also becoming more common. Several research contributions propose middleware architectures for virtualization of quantum resources, integration of scheduling jobs and uniform access to heterogeneous quantum devices.

The quantum HPC middleware frameworks introduce intermediate representations and orchestration layers that manage the quantum workload together with classical computation. While these systems have been successful in abstracting hardware heterogeneity, the selection strategy for backend is commonly reduced to simple heuristics based on device availability, qubit capacity or queue length. Deep awareness of circuit level characteristics such as gate composition or depth which are important determinants of execution fidelity on noisy quantum hardware is generally lacking [2]. This points to a gap in the need not only for hardware agnostic but also circuit aware middleware.

There is a significant body of work in quantum compilation illustrating the structure of a circuit that significantly impacts execution performance on targeted hardware. Noise aware compilation, device specific gate mapping, and topology aware routing are some of the many different approaches to reducing the accumulation of errors through adapting the circuit to hardware constraints [4]. Prior studies have identified circuit depth and two qubit gate density as key predictors of noise sensitivity and output fidelity for near term quantum devices. Latest methods use machine learning and statistical modeling to predict the performance of circuits on different backend. Learning to rank and regression based models estimate expected fidelity or execution cost by correlating circuit features with backend noise parameters. These approaches validate that joint circuit backend feature vectors can be used to estimate suitability. However, such techniques are typically applied at the compilation or mapping stage rather than at the platform level for backend selection across multiple providers and simulators.

IBM Quantum, Qiskit Runtime, and other mainstream quantum software platforms enable users to request backend properties and filter available devices according to very simple requirements using their APIs [5]. Some common utilities would be to select a backend with

a sufficient number of qubits, select operational devices and send work to devices that correspond to the “least busy” backend. However, these solutions ignore circuit depth, native gate support and noise sensitivity requirements. Community conversations, as well as user studies, confirm that trial and error methodology is used by developers as a means of finding backend, resulting in abnormal use of computational cycles. Qniverse provides various simulators and quantum devices via a common interface, though without a means of automatically suggesting a backend, circuit awareness, when there is a growing number of available backend [6].

3 Methodology

Hardware Agnostic Adaptation proposes a middleware transition layer between user submitted circuits and execution backend. The proposed layer examines all user submitted circuits and inquires about backend capabilities, filters out infeasible alternatives and identifies a list of candidate backend according to objectives such as latency, accuracy and costs.

3.1 Circuit Analysis Engine

Circuit Analysis Engine is the one that translates the quantum circuit description to a formal structure amenable to automatic reasoning. Quantum circuits described in OpenQASM languages, Qiskit or Cirq pass the analysis where the basic parameters of the circuit are identified, namely the circuit size in qubits, the maximum depth of the circuit, qubit gate decomposition of the circuit and the entanglement rate of the circuit. All these parameters together constitute the feature vector that describes the resource demand and the noise robustness of the circuit [5].

3.2 Backend Capability Mapping

The Backend Capability Mapping is responsible for keeping a constantly up to date list of all simulators and quantum processors that are currently available in the Qniverse ecosystem. The backend is represented by a capability vector that includes things like qubit capacity, connectivity requirements, native gate sets, error rates, queue latency models, cost models and computational infrastructure (CPU, GPU, QPU, FPGA) and some other parameters [6].

3.3 Intelligent Backend Recommendation Algorithm

The Intelligent Backend Recommendation Algorithm is the deciding backbone of the architecture. The algorithm takes the circuit feature vector and the backend capability vector as input and calculates the scores of each possible backend. It assesses the following parameters: predicted execution latency, fidelity in the presence of noise, cost effectiveness and resource allocation. Native gate awareness is directly taken into consideration in the algorithm; this is done through the promotion of backend with Native support of the circuit’s primary gates while demoting the backend with large amounts of decomposition of the desired gates. The scores are calculated through weighted parameters.

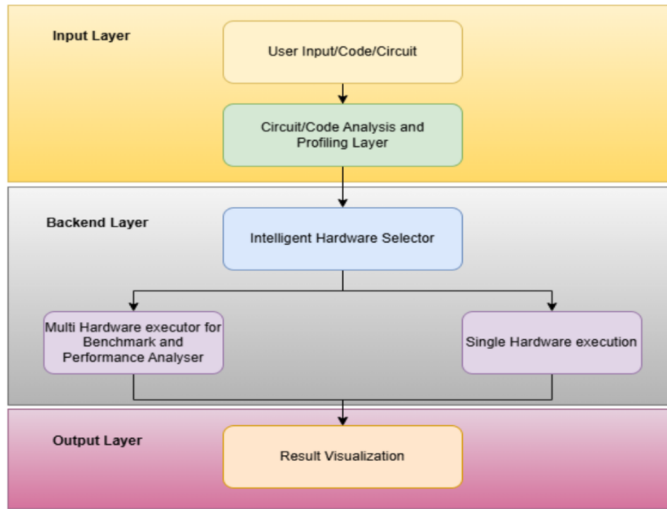


Figure 1: Architecture diagram for Intelligent Backend recommendation

Figure 1. Illustrates the architecture diagram for intelligent backend recommendation where input layer contains input section from the user and circuit, code profiling, backend layer contains intelligent hardware selector (multi hardware selector or single hardware execution) and output layer contains result visualization.

4 Algorithm

4.1 Circuit Feature Extraction Logic

This feature vector is generated once the user inputs a quantum circuit, followed by analysis of the quantum circuit's properties. The input circuit is parsed in order to find the total number of qubits required, followed by calculation of the circuit depth based on the complexity of the circuit over time. The complexity of the circuit is calculated by computing the number of single qubit gates and multi qubits gates present in the circuit, with special focus placed upon the number of two qubit gates, considering their prominent influence upon the noise sensitivity of the quantum circuit. Other properties include computational requirements defined by the need for simulation of the circuit using the state vector simulation method or the density matrix simulation method.

The Circuit Analysis Engine is capable of analyzing circuits written in OpenQASM2.0 framework functions and extracts a feature vector that is of compact size: number of qubits, circuit depth, gate distribution (qubit gates), and two qubit gate density.

This feature vector represents the circuit's computational and noise sensitivity profile.

Algorithm 1 Resource-Aware Backend Selection

Require: Quantum circuit C , backend set B **Ensure:** Backend list R

```

1: Extract circuit features  $F$  (number of qubits, circuit depth, gate counts)
2:  $R \leftarrow \emptyset$ 
3: for each backend  $b \in B$  do
4:   if  $b.max\_qubits < F.n\_qubits$  then
5:     continue
6:   end if
7:   Initialize score  $S_b$ 
8:   Evaluate qubit capacity compatibility
9:   Evaluate circuit depth with backend characteristics
10:  Evaluate memory and execution constraints
11:  Add  $(b, S_b)$  to  $R$ 
12: end for
13: Sort  $R$  in descending order of score  $S_b$ 
14: return  $R$ 

```

The algorithm 1 extracts circuit features and based on that features a score is calculated. Using this score, the backend recommendation is generated.

4.2 Backend Ranking and Scoring Logic

Based on the circuit feature vector, the system analyzes all available backends. Backends that cannot pass some initial feasibility analysis, due to absence of sufficient qubit resources, are ruled out. For the remaining backend candidate sets, the strategy calculates compatibility metrics based on how well the capabilities of the backend might satisfy the requirements of the circuit. Compatibility for native gates is investigated in an attempt to estimate the overhead of decompositions, as well as circuit depth matching the noise properties and execution models of the backend. These cost functions, queue latencies and expected fidelities are aggregated through weighted selections to produce a final list of backend ranks based on overall fit, which can either be automatically selected or offered to the user with background explanations. Capability vector is used to describe each backend, including: maximum qubit capacity, native gate set, noise models, queue latency and cost model.

Native gate is explicitly encoded at this stage by comparing circuit gate sets with backend supported native gates.

Algorithm 2 Native Gate-Aware Backend Refinement

Require: Backends R , circuit features F **Ensure:** Optimal backend b^*

```

1: for each backend  $b \in R$  do
2:   Compute noise score  $N_b$ 
3:   Evaluate native gate compatibility with circuit  $F$ 
4:   Update backend score  $S_b \leftarrow S_b + N_b$ 
5: end for
6:  $b^* \leftarrow \arg \max_{b \in R} S_b$ 
7: return  $b^*$ 

```

Algorithm 2 directly utilizes native gate awareness by penalizing backends that require excessive gate decomposition, thereby reducing expected fidelity.

5 Result and Discussion

The hardware agnostic backend adaptation module is implemented in a multi backend quantum execution platform and experimented with quantum circuits with varying numbers, depths and gates decompositions. The Circuit Analysis Engine was able to determine structural properties of the circuits accurately and helped in making feasible filtering through Backend Capability Mapping module.

The Backend Recommendation Algorithm delivered informative backend recommendations based on the compatibility, performance and use of resources. Much easier to execute circuits were more often suggested to simulator backends with smaller depths and smaller qubit counts whereas circuits with more complexities or higher noise sensitivities were directed to more suited backend environments. These observations made it certainly clear that the recommendation mechanism itself dynamically scaled to the properties of circuits and the capabilities of the backends.



Figure 2: Result containing all the backend with their scores

The integration of the hardware independent adaptation layer allowed improving the efficiency of the workflow, as the set of selection or auto selection of the back end and the reduction of trial and error testing was reduced. These results suggest that the suggested module enables the efficient, scalable and circuit-sensitive choice of the backend, thus increasing the general usability and reliability of execution in a heterogeneous quantum computing environment.

6 Conclusion

In this paper, an adaptation layer for the backend has been designed for multi backend quantum computing platforms. This adaptation layer has been designed to overcome the drawbacks of manual backend choice. By using a circuit aware middle layer technique, it allows for automated backend suggestions in relation to circuit properties and backend support instead of making an informed decision manually. This technique combines circuit assessment, backend support mapping and multi criteria rating for backend assessment in terms of latency measures, fidelity, cost and resource requirements. This method allows for better scalability in terms of evolving backend support options available for quantum computing. In general, the designed adaptation layer provides better accessibility for users who have limited understanding of backend support along with maintaining flexibility through adjustable policies.

Acknowledgement

The research work majorly focuses on C-DAC's Qniverse. A sincere gratitude and thanks is submitted to all the employees of C-DAC, NQM and Qniverse team.

References

- [1] Proctor, T., Young, K., Baczewski, A.D. et al. Benchmarking quantum computers. *Nat Rev Phys* 7, 105–118 (2025). <https://doi.org/10.1038/s42254-024-00796-z>
- [2] Preskill, J. Quantum Computing in the NISQ era and beyond. *Quantum* 2, 79 (2018). <https://doi.org/10.22331/q-2018-08-06-79>
- [3] Cirq Development Team. Cirq: A Python Framework for Creating, Editing, and Invoking Noisy Intermediate Scale Quantum (NISQ) Circuits. Google Quantum AI (2024). <https://quantumai.google/cirq>
- [4] Zhu, C., Wu, X., Yang, Z., Wang, J., Wu, A., Zheng, S., and Wang, X. Quantum Compiler Design for Qubit Mapping and Routing: A Cross-Architectural Survey of Superconducting, Trapped-Ion, and Neutral Atom Systems. *arXiv preprint arXiv:2505.16891* (2025). <https://doi.org/10.48550/arXiv.2505.16891>
- [5] Aleksandrowicz, G., Alexander, T., Barkoutsos, P., Bello, L., Ben-Haim, Y., Bucher, D., Cabrera-Hernández, F. J., et al. Qiskit: An Open-source Framework for Quantum Computing. *Zenodo* (2019). <https://doi.org/10.5281/zenodo.2562111>
- [6] B., A., Sukumar, S. H., J., S., and Raghava, N. Unlocking the Quantum Realm: Qniverse, a Unified Quantum Computing Platform. *QETCI* (2025). <https://qetci.org/unlocking-the-quantum-realm-qniverse-a-unified-quantum-computing-platform>