양자 클라우드 컴퓨팅에서 조건문을 사용한 Randomized Benchmarking 가속화

최나경 [1], 한영선 [2]
[1] 국립부경대학교 인공지능융합학과 석사과정
[2] 국립부경대학교 컴퓨터·인공지능공학부 교수

choi2019@pukyong.ac.kr, youngsun@pknu.ac.kr

Accelerating Randomized Benchmarking using Conditional Statements on Quantum Cloud Computing

Choi Nagyeong[1], Han Youngsun[2]
[1]Dept. of Artificial Intelligence Convergence, Pukyong National University
[2]Dept. of Computer and Artificial Intelligence Engineering, Pukyong National University

Abstract

Randomized benchmarking (RB) techniques for measuring the error rate of quantum computers require a large number of sequences and repeated experiments for accurate error characterization. The cloud quantum computing service, IBM Quantum Platform, offered by IBM, utilizes a first-in, first-out (FIFO) job scheduler for the purpose of ensuring fair resource distribution. However, this scheduling policy is subject to a structural limitation, in that each sequence is queued and executed individually. In this paper, a method is proposed for integrating RB sequences composed of multiple sequences into a single task, with the objective of minimizing the queue waiting time. Experiments conducted on the IBM Quantum Platform demonstrated that the proposed method could effectively reduce the overall experimental time by 70.9% and decrease the pending overhead by 64.9% compared to conventional methods. Additionally, an efficient error characterization solution was proposed under the constraints of cloud quantum computing environments.

1. Introduction

Quantum computing is recognized as a transformative technology with the potential to surpass the computational capabilities of classical computers in tackling complex problems [1]. The advent of cloud-based quantum computing platforms has significantly enhanced individual accessibility to quantum computing resources [2][3]. Despite this increased accessibility, the susceptibility of quantum systems to errors remains a critical impediment for the practical application of quantum computers [4][5].

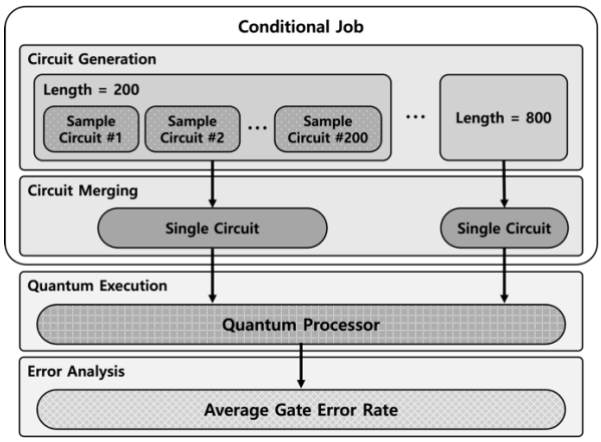
To address these limitations, the definition and measurement of errors are essential, and Randomized Benchmarking (RB) [6][7][8][9] has emerged as a prominent methodology for this purpose. RB involves applying randomly selected Clifford gates to the initial state $|0\rangle$, followed by a recovery operation to assess the coherence between the resulting and initial states, thereby yielding an average error rate. This protocol allows us to quantitatively evaluate the performance of a quantum gate applied to a target qubit by averaging the results over a large number of random sequences of varying length.

The cloud-based IBM Quantum Platform (IBMQ) uses a First-In, First-Out (FIFO) queue scheduler to ensure fair job submission [2]. This approach can introduce significant latency overhead when multiple RB sequences are submitted sequentially, as each sequence is handled as an individual job.

In this paper, a method is presented to minimize the number of jobs transmitted to the quantum processor, with the aim of mitigating factors that degrade execution efficiency within a cloud-based quantum computing environment.

2. Proposed Method: Conditional Execution

The conditional execution method proposed in this study was designed to minimize the pending between individual sequences in the queue by integrating multiple RB sequences into a single quantum circuit through conditional execution.



(Figure 1) Architecture of the proposed quantum circuit integration method.

Figure 1 illustrates the overall flow of the proposed architecture, which comprises two constituent units: a circuit-generating unit and circuit-merging unit. The operations of these two units, as depicted in Figure 1, are described in detail below.

The circuit generation unit creates circuits with Clifford gates of lengths 200, 400, 600, and 800, and their inverses for cancellation. Subsequently, ten samples are generated and grouped for each length, and the circuit-merging unit merges individual sequences into what is referred to as a single integrated circuit, capable of processing multiple samples within one execution. During the conversion process, a Hadamard gate is applied to the control qubit to generate a superposition, and the resulting measurement value is utilized as a condition in the switch statements to execute a sample circuit for each case.

Algorithm 1 Conditional Randomized Benchmarking

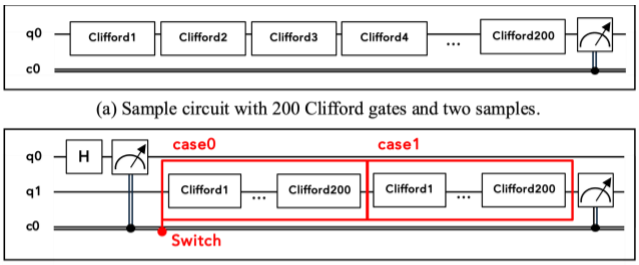
n : Number of RB samples
l : Length of each RB sequence (number of Clifford gates)
q : Size of target qubit register
s : Seed for random Clifford sequence generation

1: **procedure** CONDITIONALRB(n, l, q, s)
2: $c \leftarrow \lceil \log_2(n) \rceil$, $t \leftarrow c + q$
3: Initialize $qreg, creg, circuit(qreg, creg)$
4: **for** $i \leftarrow 0$ to $c - 1$ **do**
5: Apply H and Measure $qreg[i] \rightarrow creg[i]$
6: **end for**
7: **for** $i \leftarrow 0$ to $n - 1$ **do**
8: **if** $creg == i$ **then**
9: APPLYCLIFFORD($circuit, qreg, l, s$)
10: **end if**
11: **end for**
12: Measure $qreg[-1] \rightarrow creg[-1]$
13: **return** $circuit$
14: **end procedure**
15: **procedure** APPLYCLIFFORD($circuit, q, l, s$)
16: Set seed s, $gates \leftarrow [\,]$
17: **for** $j \leftarrow 0$ to $l - 1$ **do**
18: Append random Clifford g to $gates$
19: Apply g on $q[-1]$ ▷ Add g to circuit on $q[-1]$
20: **end for**
21: **for** g in reverse($gates$) **do**
22: Apply g^{-1} on $q[-1]$
23: **end for**
24: **end procedure**

(Algorithm 1) Quantum circuit merging algorithm for conditional execution method

As demonstrated in Algorithm 1, the conditional execution algorithm is a method that efficiently integrates multiple samples using $\lceil \log_2(n) \rceil$ control qubits, where **n** represents the number of samples. The upper bound of $\log_2(n)$ in line 2 of Algorithm 1 optimizes the number needed to process **n** samples by calculating the minimum number of control qubits required to uniquely identify **n** samples. For example, when **n=5** and $\log_2(5) \approx 2.32$, three control qubits are sufficient to differentiate between five samples using binary representations such as 000, 001, 010, and 011.

If the bit string 010 is obtained via a control qubit, only the RB sequence corresponding to index 2 is executed, whereas all other branches are skipped. This selective execution mechanism ensures that each circuit shot activates a single sample based on the measured control value.



(a) Sample circuit with 200 Clifford gates and two samples.

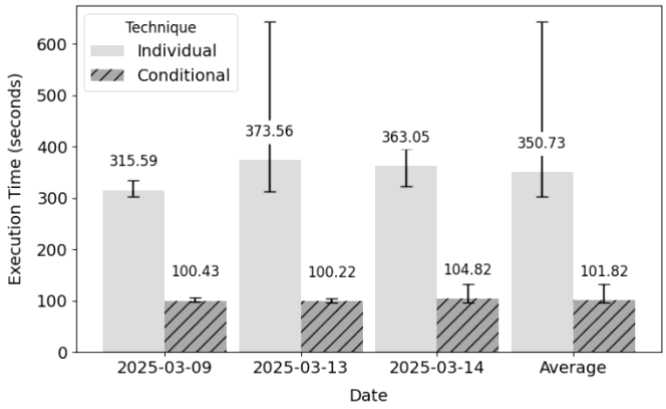
(b) Integrated circuit with 200 Clifford gates and two samples.

(Figure 2) Structural comparison between integrated and individual execution circuits for randomized benchmarking.

Figure 2 shows two implementations of a sample circuit with 200 Clifford gates. Figure 2(a) shows an example of executing a single sample using the traditional approach. This approach requires each sample to be submitted and executed as an individual job. As the number of samples increases, the number of job submissions increases proportionally, resulting in repeated job submissions. Each execution requires an independent sample, which is treated as an independent unit of execution, hereafter referred to as an individual execution. In contrast, Figure 2(b) shows the conditional execution method where two independent samples are merged into a single circuit using a single control qubit (q0). A Hadamard gate is applied to the control qubit to create a superposition, and the circuit is measured to obtain the bit string of the control qubit, which is then used to enable conditional execution. This approach allows multiple samples to be included in a single circuit and processed within a single job, as opposed to individual runs that require an individual job for each sample, reducing the number of times a single experiment requests a job. As a result, queuing and scheduling overheads are minimized.

3. Evaluation

The performance of the proposed conditional execution method was evaluated using a real quantum machine, i.e., *ibm_brussels*, provided through IBM Quantum via Qiskit [10][11]. The experiments were conducted from late February to the middle of March 2025, and all runs were independent of each other. The circuits used in the experiments were based on the RB protocol and generated ten samples each for sequences with gate lengths of 200, 400, 600, and 800 gates, with each experiment was repeated ten times.



(Figure 3) Daily average execution times (bars indicate min-max) for three representative days and the overall average, measured on the *ibm_brussels* real quantum machine via Qiskit.

To analyze performance trends under varying conditions, three representative days were selected from the experimental period and were used to construct the results presented in Figure 3. The experimental results, shown in Figure 3, indicated that the conditional execution method consistently achieved shorter execution times compared to individual execution over all three days. The average execution time was 101.82 seconds for the conditional execution method and 350.73 seconds for individual execution, resulting in approximately a 70.9% reduction in total execution time.

<Table 1> Component-wise execution time comparison on the *ibm_brussels* real quantum machine via Qiskit

Component	Individual (%)	Conditional (%)
Classical Pending	46.1	16.2
Quantum Preprocessing	22.3	32
Quantum Processing	10.9	45.3
Classical Postprocessing	20.7	6.5

As shown in Table 1, the analysis of the time taken by each step of each execution showed that the conditional execution method had a pending of 16.2%, whereas the individual execution had a pending of 46.1%, indicating that the proposed conditional execution method effectively reduced the hold in the queue.

4. Conclusion

This paper presented a conditional execution method for the queuing system of cloud-based quantum computing through the integration of conditional statement-based quantum circuits, along with experimental results. The experimental results showed that the proposed method reduced the total execution time by 70.9% compared to individual execution, and the proportion of pending in the overall execution process was reduced by 64.9%. This confirmed that quantum circuit integration could effectively minimize the overhead of individual task execution.

Acknowledgement

This research was supported by Quantum Computing based on Quantum Advantage challenge research (RS-2023-00257994) through the National Research Foundation of Korea(NRF) funded by the Korean government (MSIT).

References

[1] Roman Rietsche, et al. "Quantum Computing." Electronic Markets, 32, 4, 2525–36, 2022

[2] Gokul Subramanian Ravi, et al. "Quantum Computing in the Cloud: Analyzing job and machine characteristics." 2021 IEEE International Symposium on Workload Characterization, 39-50, 2021.

[3] Ryan LaRose, "Overview and Comparison of Gate Level Quantum Software Platforms." Quantum, 3, 130, 2019

[4] Sergio Boixo, et al. "Characterizing quantum supremacy in near-term devices." Nature Phys, 14, 595–600, 2018.

[5] John Preskill, "Quantum Computing in the NISQ era and beyond." Quantum, 2, 79, 2018.

[6] Easwar Magesan, et al. "Scalable and Robust Randomized Benchmarking of Quantum Processes." Physical Review A, 106, 18, 180504, 2011.

[7] Easwar Magesan, et al. "Characterizing quantum gates via randomized benchmarking." Physical Review A, 85, 4, 042311, 2012.

[8] Hashagen, A. K. et al. "Real Randomized Benchmarking." Quantum, 2, 85, 2018.

[9] Jonas Helsen, et al. "A new class of efficient randomized benchmarking protocols." npj Quantum Information, 5, 2019.

[10] Gadi Aleksandrowicz, et al. "Qiskit: An Open-source Framework for Quantum Computing." Zendo, 2019.

[11] Ali javadi-Abhari, et al. "Quantum computing with Qiskit." arXiv, 2024