A Piece of QAICCC: Towards a Countermeasure Against Crosstalk Attacks in Quantum Servers

Yoann Marquer
University of Luxembourg
Luxembourg
yoann.marquer@uni.lu, 0000-0002-4607-967X

Domenico Bianculli
University of Luxembourg
Luxembourg
domenico.bianculli@uni.lu, 0000-0002-4854-685X

***Abstract*—Quantum computing, while allowing for processing information exponentially faster than classical computing, is far from becoming a personal commodity. Indeed, because of cost and needs for ultra-cold temperature, shielded environment, and complex wiring for control, access to quantum computers is likely to be provided remotely, through quantum servers. Sharing quantum hardware between multiple users enables the efficient use of quantum resources, but makes some security threats possible. For instance, previous studies demonstrated that crosstalk between attacker and victim's qubits can be exploited to mount security attacks.**

In this work, we address the problem of securing user circuits submitted to quantum servers. We propose the Qubit Allocation for Inter-Circuit Crosstalk Countermeasure (QAICCC), which aims to maximize qubit usage, reduce inter-circuit crosstalk as well as intra-circuit noise. QAICCC performs (1) a crosstalk analysis of the targeted quantum processor to determine qubits involved in crosstalk with the largest intensity; (2) the allocation of qubits in the safest possible way; and (3) the application of existing techniques to further reduce crosstalk between circuits and noise in user circuits. The main contribution is the allocation algorithm, which aims to maximize qubit usage (including making unused qubits available for future usage) while minimizing the largest inter-circuit crosstalk error rate. Thus, QAICCC will support quantum computing adoption by securing the usage of quantum servers by a large number of actors.

***Index Terms*—Crosstalk analysis, Quantum server, Qubit allocation, Noise reduction, Security threat, Transpilation.**

I. Introduction

Ash-Saki et al. [1] demonstrated that crosstalk between qubits allocated to different users can be exploited for security attacks. In their attack scenario, a user (the victim) requests qubits to the server; then, another user (the attacker) requests several copies of small quantum circuits to control the largest number possible of remaining qubits. The victim runs her circuit to obtain a result in a qubit output while the attacker runs her circuit(s). Due to crosstalk from attacker's qubits to the victim's output qubit, the magnitude of the victim's expected output can be reduced below the magnitude of other results, thus tampering the victim's output. We call *crosstalk attacks* security attacks made possible by crosstalk, like Ash-Saki et al. [1]'s fault injection attack.

As NISQ computers do not have enough qubits for error correction, they resort to noisy computations that hinder quantum computing (QC) adoption. This situation warrants 1) increasing the number of available qubits and 2) reducing noise in quantum computations.

For the former, since each qubit is a precious resource that should not be wasted, we see *maximizing qubit usage* as a priority. For the latter, since crosstalk attacks are possible between users of the same quantum servers, in this paper we distinguish two kinds of noise: *inter-circuit crosstalk*, i.e., crosstalk between qubits allocated to different users, and *intra-circuit noise*, i.e., crosstalk and decoherence involving qubits allocated to the same user.

Another priority for QC adoption is that quantum servers can be used in a safe way, without a user being able to interfere with other users' executions. Hence the need to *reduce inter-circuit crosstalk*. More precisely, since crosstalk is quantified using error rates and the targeted processor may exhibit many qubit combinations leading to crosstalk, we aim to minimize the largest inter-circuit crosstalk error rate.

Finally, maximizing qubit usage while reducing inter-circuit crosstalk implies that each combination of qubits involved in crosstalk should be controlled, when possible, by the same user. Such a qubit allocation would tend to increase intra-circuit noise, which should be reduced as well to improve success rate and thus quality of service. Therefore, *intra-circuit noise reduction* is another priority in NISQ computers.

To summarize, our approach for inter-circuit crosstalk countermeasure aims to achieve the following goals, in decreasing priority:

G1: maximizing qubit usage (including making unused qubits available for future usage);
G2: minimizing the largest inter-circuit crosstalk error rate;
G3: reducing intra-circuit noise (crosstalk and decoherence).

II. Approach

Our Qubit Allocation for Inter-Circuit Crosstalk Countermeasure (QAICCC) approach takes as input the targeted quantum processor, user circuits, and (potential) trusted users; it returns transpiled circuits that can be executed in a safe way, minimizing the threat of crosstalk attacks.

The approach consists of four main steps. In Step 1, the processor is analyzed to determine crosstalk error rates (§ II-A). In Step 2, qubits are allocated to users (§ II-B). In Step 3, user circuits are transpiled according to the selected qubit allocation to match the processor connectivity and supported gates (§ II-C). In Step 4, transpiled circuits are transformed to reduce noise during quantum executions (§ II-D).

A. Step 1: Crosstalk analysis

In this step, the processor is analyzed (e.g., with pyGSTi [2]) to determine crosstalk error rates between qubits. We consider the impact of one or two qubits on another qubit (as in Ash-Saki et al. [1]'s work) and the impact of two qubits on two qubits (i.e., gate errors, as in Murali et al. [3]'s work), which we respectively call 1-1, 2-1, and 2-2 crosstalk.

We take inspiration from Murali et al. [3]'s methodology by considering only neighboring qubits. More precisely, each m-n crosstalk is measured so that the $m+n$ considered qubits form a connected component.

In QAICCC, crosstalk is quantified using a single metric like the *composite score*, which is the sum of the Stochastic error rates and the square of the Hamiltonian error rates, as defined in Ash-Saki et al. [1]'s work.

B. Step 2: Allocation algorithm

In this step, we allocate qubits to users in order to meet G1 and G2, using an *allocation algorithm* detailed in the long version of this work [4]. The algorithm takes as input the connectivity of the processor, the information on the size of the user circuits, and the error rates returned by Step 1.

In case the number of qubits to allocate is smaller than the number of available qubits, to meet G1, the algorithm allocates qubits for potential, future users, by introducing an untrusted *idle user* requesting for the rest of the qubits. To ensure user circuits can be transpiled using the obtained qubit allocations, each user's allocation is determined so that it forms a connected component. Thus, the unused qubits will be connected, which increases the chance of being able to allocate them to new users.

The algorithm takes inspiration from genetic search by maintaining an evolving population of possible allocations, but does not involve any randomness. To meet G2, it processes error rates in decreasing order of score, so that the qubits involved in crosstalk with the largest intensity are allocated first. For a given error rate, each allocation in the population is tested to determine if it is already safe, i.e., impacting qubits are allocated to trusted users or impacted users control impacting qubits. If the allocation is safe, then it is used in the next iteration. Otherwise, it is removed from the population, and then (when possible, due to size constraints) new safe allocations are generated from it (based on the processor connectivity and circuit sizes) and added to the population. In this way, the last individuals in the population correspond to allocations of qubits done in the safest possible way.

C. Step 3: Transpilation

In this step, user circuits are transpiled according to the selected qubit allocation. Allocations returned by Step 2 are sorted by increasing score, so the allocations obtained last (i.e., those addressing the largest number of error rates) are the first ones. Starting from the first allocation, an allocation is selected and tested as follows. Qiskit is called to transpile the user circuits according to the current allocation, to match the processor connectivity and supported gates. If the transpilation is successful, then the transpiled circuits are selected for Step 4. Otherwise, the next allocation is selected and tested. This process continues until an allocation is successful. If no allocation is successful, then QAICCC returns an error.

D. Step 4: Noise reduction

In this step, QAICCC uses existing techniques like the XtalkSched scheduler [3] and ColorDynamic [5] on the transpiled circuits to reduce noise further. To meet G2, the priority is to reduce first remaining inter-circuit crosstalk of the qubit allocation selected in Step 3. To meet G3, these techniques can be also independently applied to each circuit, to reduce intra-circuit noise.

III. Conclusion and future work

We have presented QAICCC, an approach for securing users' program executions from crosstalk attacks in quantum servers. Our ongoing work focuses on implementing QAICCC (in Python, for better integration with frameworks like Qiskit and PennyLane), followed by an experimental campaign. We plan to evaluate the qubit allocation algorithm by comparing the largest inter-circuit crosstalk error rate between its allocation and an allocation performed by the last version of Qiskit; we plan to evaluate QAICCC by simulating various combinations of attacker, victim, trusted, and untrusted users and comparing—in terms of success rate—the execution of QAICCC's transpiled circuits with transpiled circuits obtained by baselines like Qiskit and XtalkSched [3] alone.

Acknowledgment

This project has received funding from SES and the Luxembourg National Research Fund under the Industrial Partnership Block Grant (IPBG), ref. IPBG19/14016225/INSTRUCT.

References

[1] A. Ash-Saki, M. Alam, and S. Ghosh, "Analysis of crosstalk in nisq devices and security implications in multi-programming regime," in *Proceedings of ISLPED '20*. ACM.

[2] pyGSTi, "A python implementation of gate set tomography," 0.9.13. [Online]. Available: https://github.com/sandialabs/pyGSTi

[3] P. Murali, D. C. Mckay, M. Martonosi, and A. Javadi-Abhari, "Software mitigation of crosstalk on noisy intermediate-scale quantum computers," in *Proceedings of ASPLOS '20*. ACM.

[4] Y. Marquer and D. Bianculli, "A Piece of QAICCC: Towards a Countermeasure Against Crosstalk Attacks in Quantum Servers," 2025. [Online]. Available: https://orbilu.uni.lu/handle/10993/64693

[5] Y. Ding, P. Gokhale, S. F. Lin, R. Rines, T. Propson, and F. T. Chong, "Systematic crosstalk mitigation for superconducting qubits via frequency-aware compilation," in *2020 MICRO*. IEEE, Oct 2020, pp. 201–214.