Distributing Quantum Programs Shot by Shot

Giuseppe Bisicchia*
Department of Computer Science
University of Pisa, Pisa, Italy
giuseppe.bisicchia@phd.unipi.it

Jose Garcia-Alonso
Quercus Software Engineering Group
University of Extremadura, Cáceres, Spain
jgaralo@unex.es

Antonio Brogi
Department of Computer Science
University of Pisa, Pisa, Italy
antonio.brogi@unipi.it

Juan M. Murillo
Quercus Software Engineering Group
University of Extremadura, Cáceres, Spain
juanmamu@unex.es

***Abstract*—Despite all the efforts and the continuous and exciting daily achievements that are being made, currently available Quantum Process Units (QPUs) still offer few qubits and are extremely noise-sensitive. In spite of these limitations, researchers are taking advantage of the possibilities offered by these QPUs and trying to maximise their use. Two different dimensions are the major focus of the current scientific work in the area. Some works try to optimise the Quantum *load* by distributing different programs on different QPUs. Other works try to optimise the execution of a single Quantum circuit, by cutting it into smaller *fragments*, making it possible to execute circuits that are otherwise theoretically impractical for current QPUs. In this work, we discuss a different dimension of the distribution of quantum programs. Indeed, due to the probabilistic nature of Quantum Mechanics, it is usually required to iterate the execution of a Quantum program numerous times (called *shots*). Leveraging this constraint, we propose taking into account the *shots* as an additional dimension. We suggest distributing the shots of a Quantum program among multiple independent QPUs. By fully exploiting the qualities of different QPUs, such behaviour can enable more fine-grained management of the requirements of a Quantum application. Additionally, multi-QPU executions significantly increase the resilience of a Quantum program execution to QPU failures. Finally, such an approach is very suitable to be fully customised by users. We also propose a Domain Specific Language which allows users to specify their needs as distribution policies, in a simpler way than with currently available hard-coded solutions.**

***Index Terms*—Quantum Software Engineering, Hybrid Classical-Quantum Services, Distributed Quantum Computing**

POSTER RELEVANCE

This work fits in the field of *Distributed Quantum Computing*, a currently *hot-topic* in Quantum Computing. Indeed, because of the very severe restrictions currently featured by the available Quantum Process Units (QPUs), as well as the very high heterogeneity and the lack of standards on the design and development of QPUs and Quantum compilers, researchers and practitioners are trying to make the most of the contemporary scenario.

And it is in this context that numerous techniques are rising to exploit the Quantum offer at its best. Such approaches focus on either distributing the *load*, generated by the parallel submission of Quantum programs, by sending on different QPUs different Quantum programs or on distributing a single large Quantum program (thus impossible to execute in current QPUs) by cutting it in smaller *fragments* actually executable.

The relevance of our methodology resides in proposing a different point of view on the whole problem, by suggesting to consider a further dimension among which to perform the distribution, and by which to get high *resilience*, *expressiveness* and *fine-grained management* of the distribution decisions.

Given a Quantum program (which can also be a *fragment* cut by a larger program), we propose to distribute the shots of such a program, in such a way that different QPUs can be asked to execute a certain amount of shots of a single Quantum Program. Distributing the shots offers fine-grained management of the execution performance of a given Quantum program, making it possible to exploit the performance of various QPUs.

Such a distribution methodology can enable a higher resilience to the failures of (possibly many) QPUs. Exploiting multiple QPUs, if one (or some) of them fails, the execution of the shots on the other ones can still suffice to obtain useful results. Furthermore, we will show that distribution policies on shots can be easily expressed and encoded. Indeed, we also propose a Domain Specific Language with which users can express constraints on the QPUs, compilers (and/or combinations of them) to use and metrics to rank the possible distributions. So our methodology is fully customisable and users can express their own requirements and desires in a simple and effective way.

Concluding, we believe that our proposal can be relevant to the field of Quantum Computing and more in detail Quantum Software Engineering and Distributed Quantum Computing, offering an innovative point of view on a hot problem that can raise interesting scientific discussions and (hopefully) innovations.

* Corresponding author.

Distributing Quantum Programs Shot by Shot*

Giuseppe Bisicchia
Department of Computer Science
University of Pisa, Pisa, Italy
giuseppe.bisicchia@phd.unipi.it

Jose Garcia-Alonso
Quercus Software Engineering Group
University of Extremadura, Cáceres, Spain
jgaralo@unex.es

Antonio Brogi
Department of Computer Science
University of Pisa, Pisa, Italy
antonio.brogi@unipi.it

Juan M. Murillo
Quercus Software Engineering Group
University of Extremadura, Cáceres, Spain
juanmamu@unex.es

***Abstract*—Despite all the efforts and the continuous and exciting daily achievements that are being made, currently available Quantum Process Units (QPUs) still offer few qubits and are extremely noise-sensitive. In spite of these limitations, researchers are taking advantage of the possibilities offered by these QPUs and trying to maximise their use. Two different dimensions are the major focus of the current scientific work in the area. Some works try to optimise the Quantum *load* by distributing different programs on different QPUs. Other works try to optimise the execution of a single Quantum circuit, by cutting it into smaller *fragments*, making it possible to execute circuits that are otherwise theoretically impractical for current QPUs. In this work, we discuss a different dimension of the distribution of quantum programs. Indeed, due to the probabilistic nature of Quantum Mechanics, it is usually required to iterate the execution of a Quantum program numerous times (called *shots*). Leveraging this constraint, we propose taking into account the *shots* as an additional dimension. We suggest distributing the shots of a Quantum program among multiple independent QPUs. By fully exploiting the qualities of different QPUs, such behaviour can enable more fine-grained management of the requirements of a Quantum application. Additionally, multi-QPU executions significantly increase the resilience of a Quantum program execution to QPU failures. Finally, such an approach is very suitable to be fully customised by users. We also propose a Domain Specific Language which allows users to specify their needs as distribution policies, in a simpler way than with currently available hard-coded solutions.**

***Index Terms*—Quantum Software Engineering, Hybrid Classical-Quantum Services, Distributed Quantum Computing**

I. Introduction

Nowadays, much research is being done on better, more efficient Quantum Process Units (QPUs). However, current QPUs still are very noise-sensitive and feature only a small amount of qubits. These limitations strongly restrict the computation that can be effectively done on a QPU. But, while scientists and engineers are working on producing better QPUs, other researchers are trying to figure out how to make the best use of existing available QPUs.

One very active research line is that of *Distributed Quantum Computing*. More in detail, there are two main ways to actuate such distribution. It is possible to distribute the Quantum *load*, by sending different Quantum programs to different QPUs [1], [2], or it is possible to cut bigger infeasible Quantum programs into smaller *fragments* that can be performed on current QPUs [3], [4].

In this work, we propose a change of perspective from the state-of-the-art on *Distributed Quantum Computing*. Indeed, we suggest that distributing the *shots* of Quantum programs (that can also be circuits *fragments*) among multiple QPUs can (1) offer a *fine-grained management* of Quantum programs requirements and available QPUs, (2) improve the *resilience* of Quantum programs execution to QPUs failures, and (3) enable a more customisable and *expressive* methodology for the users that can declaratively express their requirements through a Domain Specific Language (DSL).

II. Background

The problem of the distribution of Quantum programs is currently a hot topic in Quantum Computing. There are two main lines of attack researchers are following.

Various works try to optimise the *load*, by sending different Quantum programs to different QPUs. The *Quantum API Gateway* [1], for instance, selects the best QPU most suitable for a given algorithm considering its architecture, the circuit's width and cost and time requirements. The *NISQ Analyzer* [2] selects the best combination of QPU and circuit implementation considering the circuit's input, width and depth and the SDK used. Several extensions have been proposed (e.g., [5] and [6]). [7] and [8] propose two quantum job schedulers both reasoning on the estimated fidelity and waiting times. Finally, in [9], the best combination of QPU, compiler and compiler options are selected, through Machine learning, reasoning on the gate and measurement operations *fidelity*.

* This work is supported by the QSALUD project (EXP 00135977 / MIG-20201059) in the lines of action of the Center for the Development of Industrial Technology (CDTI); and by the Ministry of Economic Affairs and Digital Transformation of the Spanish Government through the Quantum ENIA project call – Quantum Spain project, by the European Union through the Recovery, Transformation and Resilience Plan – NextGenerationEU within the framework of the Digital Spain 2025 Agenda, and by UNIPI PRA 2022 64 "hOlistic Sustainable Management of distributed softWARE systems" (OSMWARE) project funded by the University of Pisa, Italy. This work is also part of the Grant PID2021-1240454OB-C31 funded by MCIN /AEI /10.13039/50100011033 and by "ERDF A way of making Europe".

The other popular research line is based on the idea of cutting large Quantum circuits in smaller *fragments* that are actually executable on the available QPUs. *CutQC* [3], for instance, features a hybrid approach that cuts Quantum circuits a distributes them onto quantum (i.e., QPU) and classical (i.e., CPU or GPU) platforms for co-processing. [4], instead, employs randomised measurements and classical communication to coordinate measurement outcomes and state preparation. Finally, with a different approach, [10] proposes to distribute a Quantum circuit over a homogeneous network of QPUs minimising the quantum communication cost.

To the best of our knowledge, our proposal is the first to tackle the problem of distributing quantum programs, across multiple QPUs, with a *shot-by-shot* approach. Furthermore, our work is also the first to enable users to fully customise their distribution policies, instead of relying on hard-coded solutions.

III. Shot by Shot Distribution

The rationale behind our proposal is that a user-provided circuit is. first compiled with different compilers and optimised for different QPUs, thus generating a set of actually executable compiled Quantum circuits. Such circuits, together with the updated status of the available QPUs are given in input to a *reasoner* which, following the user requirements, generates a final *dispatch* (i.e., a set of triples each of them composed by a QPU, a compiled circuit and a number of shots). Such *dispatch* can eventually be executed by interacting with the Quantum providers.

Relying on multiple QPUs, each of them executing only a fraction of the shots, guarantees that if one (or even more) of that QPUs fail then the other ones can still complete the execution of their shots (being completely independent), still ensuring useful results.

Furthermore, reasoning *shot-by-shot* enables a high level of *fine-grained* management of the circuit's requirements and the QPUs, working with the very minimum unit of computation and thus possibly exploiting different QPUs with different performance to have at the end global results that leverage on such diversity to (possibly) increase the quality of the results.

IV. Expressing Requirements Declaratively

Users can easily customise the distribution process by expressing their requirements. With that aim, we propose a DSL which relies on two main concepts to specify such desires and needs. Developers can express their objectives for the final dispatch by defining, declaratively, a set of *metrics* and *constraints* (e.g., through a set of logic rules and predicates).

Metrics allows users to rank the final dispatches when multiple of them are available. In such a case developers can tell the reasoner how to order them by selecting the dispatches that better optimise the metrics.

Constraints, instead, specify when a specific QPU and/or compiled circuit (and/or combination of them) are accepted and when instead must be discarded. Additionally, it is also possible to express constraints on when a final dispatch is accepted or not. Finally, it is possible to specify constraints on the metrics (or combinations of them).

Such a language is designed to be simple to use and understand but, at the same time, powerful enough to enable users to express their desires and objectives to fits their needs, applications and working scenarios.

V. Conclusions

In this poster, we introduced an innovative point of view on the problem of distributing Quantum programs. Our proposal relies on the idea of considering distributing Quantum programs (which can also be fragments cut by a larger Quantum circuit) through a new dimension. Given a Quantum program, we suggest distributing individually each shot of that program based on a particular distribution policy (possibly) expressed directly by the user through a DSL.

Such behaviour also enables a higher level of resilience to QPUs failures, being able to have useful results also when one (or many) QPUs fail, by relying upon the good ones.

Our proposal offers fine-grained management of Quantum programs by distributing the shots among multiple QPUs. Such QPUs will feature different and peculiar characteristics and performances, therefore, with our methodology will be possible to better exploit such Quantum offers and to better satisfy the user requirements.

Indeed, our architecture is also accompanied by a DSL in which the developers can express declaratively *constraints* on the allowed QPUs, compilers, and combination of them as well as possible distributions. Additionally, users can express also a set of metrics on such distributions so as to rank them accordingly and always select, among the valid ones, which better fits the user needs.

Our proposal offers, then, an innovative point of view by suggesting the distribution of Quantum programs shot-by-shot, enabling a higher level of resilience, fine-grained management and customisation, also offering a DSL with which users can simply declare their requirements as distribution policies.

References

[1] J. García-Alonso *et al.*, "Quantum software as a service through a quantum API gateway," *IEEE Internet Comput.*, vol. 26, pp. 34–41, 2022.

[2] M. Salm *et al.*, "The NISQ analyzer: Automating the selection of quantum computers for quantum algorithms," in *CCIS*, vol. 1310, 2020, pp. 66–85.

[3] W. Tang *et al.*, "Cutting quantum circuits to run on quantum and classical platforms," *CoRR*, vol. abs/2205.05836, 2022.

[4] A. Lowe *et al.*, "Fast quantum circuit cutting with randomized measurements," *Quantum*, vol. 7, p. 934, 2023.

[5] M. Salm *et al.*, "Automating the comparison of quantum compilers for quantum circuits," in *CCIS*, vol. 1429, 2021, pp. 64–80.

[6] M. Salm *et al.*, "How to select quantum compilers and quantum computers before compilation," in *CLOSER*, 2023, pp. 172–183.

[7] G. S. Ravi *et al.*, "Adaptive job and resource management for the growing quantum cloud," in *IEEE QCE*, 2021, pp. 301–312.

[8] M. Grossi *et al.*, "A serverless cloud integration for quantum computing," *CoRR*, vol. abs/2107.02007, 2021.

[9] N. Quetschlich *et al.*, "Predicting good quantum circuit compilation options," *CoRR*, vol. abs/2210.08027, 2022.

[10] R. G. Sundaram *et al.*, "Efficient distribution of quantum circuits," in *LIPIcs*, vol. 209, 2021, pp. 41:1–41:20.

IEEE Quantum Week 2023

Distributing Quantum Programs Shot by Shot

Giuseppe Bisicchia[1] Jose Garcia-Alonso[2] Antonio Brogi[1] Juan M. Murillo[2]

[1]University of Pisa [2]University of Extremadura

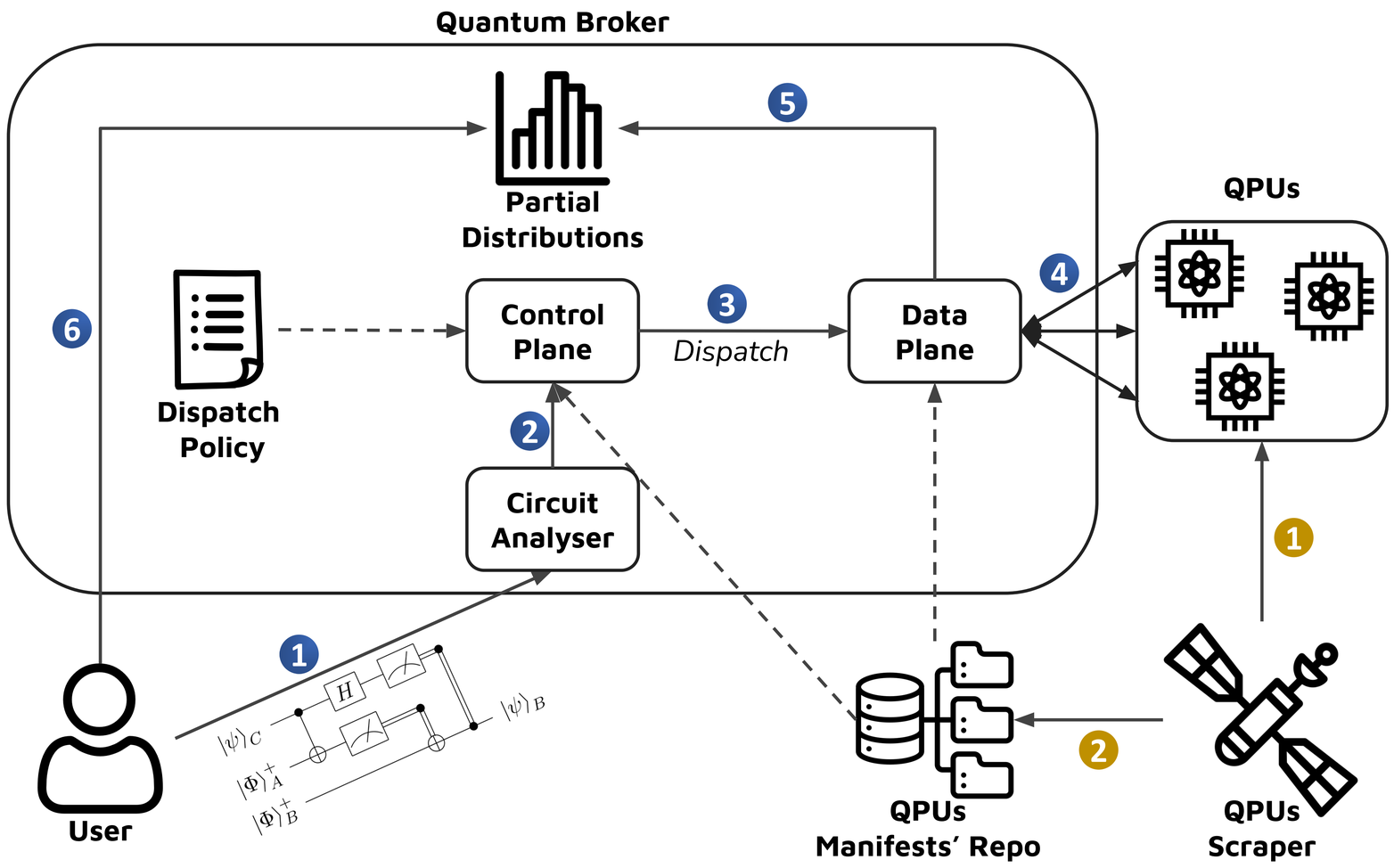
Contribution

We propose to **distribute** *Quantum Programs* across **multiple** *Quantum Process Units* (QPUs) by considering **each shot independently**

Key Advantages

1. High **resilience** to QPU **failures**
2. **Fine-grained** management
3. High **customisability**

Proposed Architecture



Control Plane

Determines the most suitable *dispatch*

Data Plane

Performs the *dispatch* and collects the outputs (*partial distributions)*

Dispatch Structure

A set of triples *(QPU, Compiled Circuit, Shots)*:

```
[(IBM Kyiv, QiskitC1, 200), (IBM Perth, QiskitC2, 300),
(Aspen-M-3, ForestC1, 750), (Aspen-M-3, ForestC2, 750)]
```

Dispatch Policies

Constraints to filter out unsuccessful dispatches:
`No simulators, only QPUs in Europe`

Metrics to rank successful dispatches:
`Execution time, cost, fidelity`

Contacts