Execution Management of Distributed Quantum Computing Jobs

Davide Ferrari, Michele Bandini and Michele Amoretti
Quantum Software Laboratory
Department of Engineering and Architecture
University of Parma
43124 Parma, Italy
{davide.ferrari1 | michele.bandini | michele.amoretti}@unipr.it

***Abstract*—In a distributed quantum computation, a large quantum circuit gets sliced into sub-circuits that must be executed at the same time on a quantum computing cluster. The interactions between the sub-circuits are usually defined in terms of non-local gates that require shared entangled pairs and classical communication between different nodes. Assuming that multiple end users submit distributed quantum computing (DQC) jobs to the cluster, an execution management problem arises. This is actually a *parallel job scheduling* problem, in which a set of jobs of varying processing times need to be scheduled on multiple machines while trying to minimize the length of the schedule. In a previous work, we started investigating the problem considering random circuits and approximating the length of each DQC job with the number of layers of the circuit. In this work, we put forward the study by considering a more realistic model for estimating DQC job lengths and by performing evaluations with circuits of practical interest.**

I. Introduction

Currently available quantum computers work with a small number of noisy qubits with non-uniform quality. To cope with large quantum circuits, a different approach consists in using distributed quantum computing (DQC) [1] to increase the number of available qubits by means of networked quantum processing units (QPUs). Both classical and quantum communications are necessary, to that purpose.

In a datacenter scenario, DQC may exploit already existing low-latency communication technologies, such as Omni-Path and InfiniBand, for high-speed classical messaging between quantum processors. Also entanglement distribution, which is highly relevant to DQC, can be achieved over standard telecom fibers connecting the QPUs [2]. DQC at geographical scale is also expected further into the future, leveraging the Quantum Internet, i.e., metropolitan-area and wide-area quantum networks that work in synergy with the existing Internet [3], [4]. In this context, long-distance entanglement is enabled by quantum repeaters [5].

A distributed quantum computation can be represented as a set of quantum programs that run in parallel and have frequent interactions, either communicating by means of quantum messages or establishing quantum entanglement between their qubits. In particular, entanglement plays a major role in the implementation of non-local gates (Figure 1).

A set of quantum programs composing a distributed quantum computation can be denoted as a *DQC job*. In the envisioned scenario, multiple DQC jobs can be concurrently submitted to several networked QPUs. Incoming DQC jobs are queued and handled by a specialized execution management service, which will be denoted as *Execution Manager* from now on.

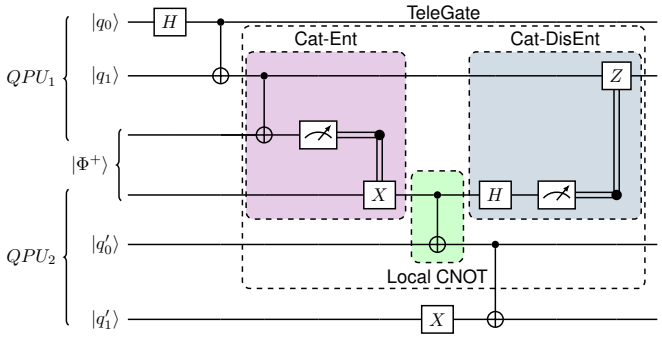

Fig. 1: Example of distributed quantum computation performed on two QPUs. The TeleGate primitive is used to implement a non-local gate (a CNOT between the second data qubit of the first QPU and the first data qubit of the second QPU).

In a previous work [6], we perfomed a preliminary investigation of the DQC job scheduling problem considering random circuits and approximating the length of each DQC job with the number of layers (i.e., the *depth*) of the circuit. In this way, it was possible to perform a preliminary assessment of two job scheduling strategies, using the makespan and two DQC-specific utilization metrics – one concerning the QPUs, the other concerning the network. One conclusion was that the makespan alone is not sufficient to compare DQC job scheduling strategies, as makespan optimality needs almost deterministic entanglement routing between QPUs, which is very hard to achieve. Another conclusion was that novel DQC-specific job scheduling algorithms are absolutely necessary.

In this work, we get further insights on this research problem by introducing a more realistic model for estimating DQC job lengths and by performing evaluations with circuits of practical interest. In particular, we consider jobs that correspond to the distributed version of GHZ circuits,

979-8-3315-4137-8/24/$31.00 ©2024 IEEE
DOI 10.1109/QCE60285.2024.10269

which are generalizations of the well-known Bell state that involve three or more qubits. Due to their extremely non-classical properties, GHZ states serve as the central resource for a number of important applications in quantum information science, including secret sharing, sensing, and fusion-based quantum computing [7].

The paper is organized as follows. In Section II, the reference DQC workflow model is illustrated in terms of modules and exchanged data. In Section III, the Execution Manager is described in detail, with particular emphasis on DQC job length estimation. In Section IV, a few relevant examples are considered for assessing the considered job scheduling policies. Finally, Section V concludes the paper with a discussion of open problems and future research directions.

II. DQC Workflow Model

In DQC, the workflow includes multiple components that work together and sequentially, as illustrated in Figure 2. First of all, a *monolithic circuit* to be executed is sent to the *Quantum Compiler* [8], which will use the information about the topology of the network and the high-level QPU features to correctly and optimally distribute the circuit between multiple QPUs. The output of the compiler is a DQC job that contains the information about which quantum operations are executed by which node and inter-node quantum and classical communications.

Each of these jobs enters a queue to be overseen by the Execution Manager, which knows in great detail both the network and the QPU features, and schedules the jobs based on current QPU and network workloads. The queue can be dealt with either on a first-in first-out basis, by simply executing the jobs as they arrive, or with more sophisticated algorithms, like the one that will be described in Section III-C.

The last part of the proposed model is the one where the performance is analyzed. Several different aspects are evaluated, starting from the correctness of a compiled circuit. The other performance indicators are described in Section IV.

III. Execution Manager

In this section, the Execution Manager is described in detail, considering the characterization of its input, the estimation of incoming jobs' length, and the job scheduling algorithms.

A. Input Characterization

The jobs in the queue are made of different *programs*, one for each QPU, that have been obtained from the compilation for distributed execution. Each job descends from a monolithic circuit, which is split into subcircuits including non-local gates whose execution relies on shared entangled pairs and classical communication.

Aside from the queue, the Execution Manager also needs to know the network topology and hardware parameters for each QPU. These parameters may include the time needed to perform qubit initialization, one-qubit and two-qubit gate execution, readout operations, classical messaging and entanglement generation.

Given these inputs, the Execution Manager will take into account - at least - the number of required QPUs, the number of qubits for each QPU, and the job length, in order to schedule jobs over the network, according to a specific scheduling algorithm.

B. DQC Job Length

Given the hardware characteristics mentioned in Section III-A, the Execution Manager can estimate the time required for each job to complete. To do this, it analyzes the programs that make up a job and obtains directed acyclic graphs, one graph for each program.

In such graphs, the vertices correspond to the tasks that an individual program must perform. These tasks can be of different types: single qubit operations, two qubit operations, entanglement generation, readout operations, waiting for another program.

The edges instead correspond to the qubits involved in the aforementioned tasks. Depending on the type of task, each edge is assigned a weight, which corresponds to the time needed to perform the task, estimated based on the input parameters. The length of a DQC job (intended as execution time) corresponds to the length of the longest weighted path within the graphs of the programs that compose the job.

Carbon initialization time	$300\ \mu s$
Electron initialization time	$2\ \mu s$
Carbon one-qubit gate duration	$20\ \mu s$
Electron one-qubit gate duration	$5\ ns$
Electron two-qubits gate duration	$500\ \mu s$
Electron readout time	$3.7\ \mu s$
Entanglement generation time (fidelity of 0.8 [9])	$0.35\ s$

TABLE I: Hardware parameters for color-centers based QPUs, taken from [9], [10].

For example, let us consider the parameters in Table I, which pertain QPUs based on color centers, as described in [9], [10]. The graphs that are necessary to estimate the execution time of a CatEnt operation (shown in Fig. 1) are depicted in Fig. 3. There is one graph for each QPU and each edge carries information about the involved qubit and the duration of the next task.

C. Job Scheduling

DQC jobs, with respect to classical jobs, possess a larger number of static properties which can be exploited by the scheduling policy. For instance, every job has fixed values of width (the number of data qubits), length, computation-to-communication ratio [6] (i.e., the ratio between the number of local gates and the number of non-local gates), and more. The Execution Manager can then accordingly make clever decisions on the order of execution of the jobs.

The *parallel job scheduling* optimization problem, which the Execution Manager deals with, has been extensively investigated [11]–[14]. It consists of a set of jobs with different processing times which need to be scheduled on m machines, while minimizing the makespan: that is, the length of the

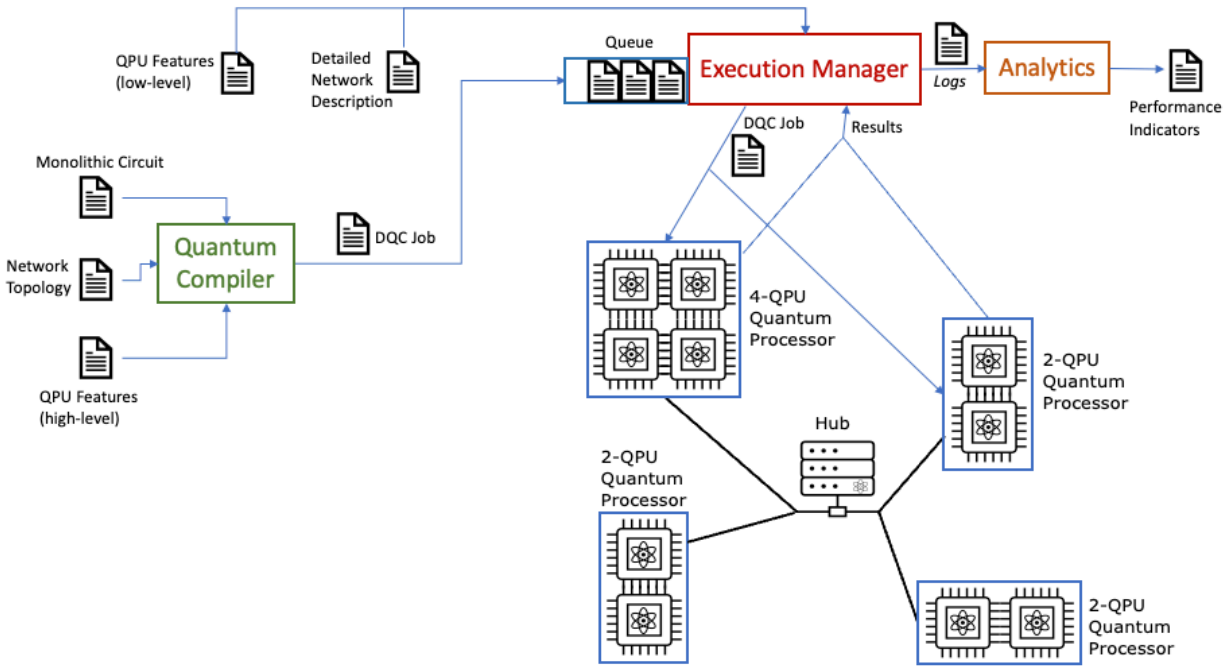

Fig. 2: The proposed DQC workflow model, whose main modules are the Quantum Compiler and the Execution Manager. The quantum network includes different quantum computing nodes connected by means of a hub, which means that the network is local (datacenter) or metropolitan. Each node is equipped with multiple QPUs.

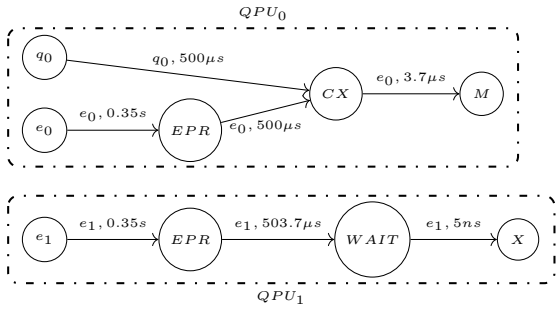

Fig. 3: Directed acyclic graph representing the tasks needed to perform a CatEnt operation across two QPUs.

schedule. Each job has a processing time p_i and requires the concurrent use of q_i machines. In general, the problem is NP-hard.

List-scheduling (LS) [11] is an efficient greedy algorithm that guarantees a makespan that is always at most $2 - 1/m$ times the optimal makespan. List-scheduling also works in the online setting where jobs arrive over time and the length of a job becomes known only when it completes [13].

In this work, LS is compared to the first-in first-out (FIFO) algorithm, for scheduling the execution of distributed quantum computations.

The LS algorithm is illustrated by Algorithm 1. The queue is explored backward, starting from the first element arrived,

Algorithm 1 List-Scheduling
Input: job queue J, idle QPU set Q
Output:

```
function SCHEDULE
    i ← 0
    while Q ≠ ∅ do
        next ← J[i]
        if ∃q ⊆ Q : q = next.q then
            schedule next
            Q ← Q\q
            J ← J\next
        else
            i ← i + 1
        end if
    end while
end function
```

until all available QPUs are assigned a job. The algorithm is executed every time a job completes. The FIFO algorithm is much more simple, as it assumes that the first job entering the queue is the first job scheduled for execution.

IV. Performance Evaluation

We evaluated the performance of different scheduling algorithms for a queue of five different jobs, to be scheduled on a network of six QPUs, each with two qubits. Jobs are characterized by different lengths and number of required QPUs q, as shown in Fig. 4.

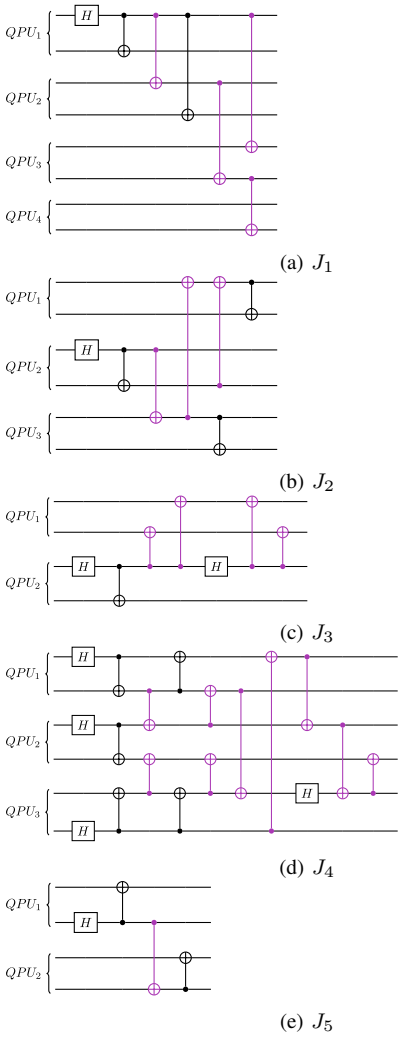

Fig. 4: DQC jobs with varying length and number of required QPUs. J_1, J_2 and J_5 are circuits used to generate GHZ states [15].

The length of each DQC job, in terms of execution time, was estimated as explained in Section III-B. For the required hardware parameters, we adopted color-centers based QPUs and took values from [9], [10], as reported in Table I.

Given the parameters in Table I, the estimated lengths of the jobs in Fig. 4 are reported in Table II.

Job	Length [s]	q
J_1	1.055	4
J_2	0.708	3
J_3	0.706	2
J_4	1.406	3
J_5	0.357	2

TABLE II: For each job, its length and number of required QPUs is reported.

The FIFO and LS algorithm were tested against three different job queues, as reported in Table III. With queue 1, the LS algorithm produces a schedule noticeably shorter than the one produced by FIFO. This can be visualized in Fig. 5a and Fig. 5b, where the different schedules that result from FIFO and LS are depicted. It should be noticed that with LS the QPUs are rarely unutilized, in contrast with FIFO. Another example worth of notice is illustrated in Fig. 5c and Fig. 5d, where LS exploits non adjacent QPUs for J_2, thus optimizing the total makespan.

	Queue	Makespan [s] FIFO	LS
1	$\{J_5, J_4, J_3, J_2, J_1\}$	3.167	2.470
2	$\{J_1, J_4, J_2, J_5, J_3\}$	2.827	2.461
3	$\{J_5, J_1, J_4, J_2, J_3\}$	2.469	2.461

TABLE III: Total makespan of schedules produced by FIFO and LS with different queues.

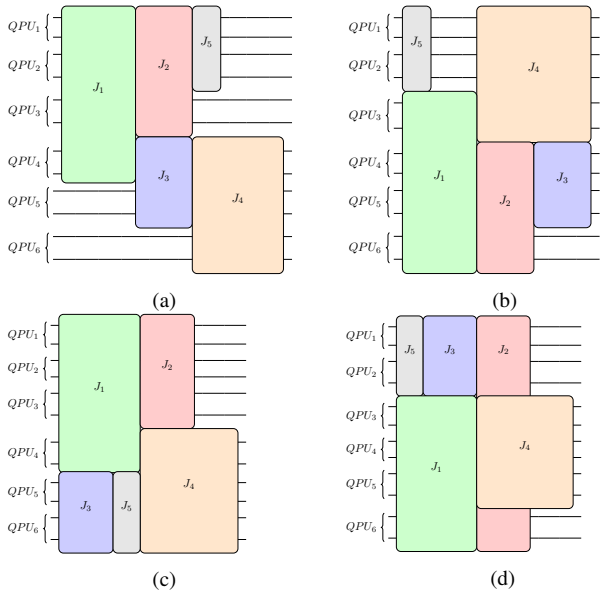

Fig. 5: Scheduling output examples: (a) FIFO scheduling of queue 1; (b) FIFO scheduling of queue 3; (c) List scheduling of queue 1; (d) List scheduling of queue 3.

To get quantitative results that go beyond the traditional concept of makespan, in the previous work [6] we introduced two metrics denoted as *QPU utilization* and the *quantum network utilization*. Those metrics assumed that the length of the jobs and the makespan could be approximated in terms number of quantum computation layers. In the following, we improve the definitions, considering estimated execution times instead.

QPU utilization is defined as the ratio between the actual workload and the potential workload given the makespan M, namely:

$$U_{\text{QPU}} = \frac{\sum_i p_i q_i}{M n_{\text{QPU}}} \in [0, 1], \tag{1}$$

where p_i is the estimated execution time of the i-th job, q_i is the number of required QPUs of the i-th job, M is the

makespan of the schedule, and n_{QPU} is the number of the system's QPUs. The shorter the makespan, the higher the QPU utilization.

Quantum network utilization is defined as the ratio between the total number of scheduled non-local gates and the maximum number of non-local gates given the makespan M, namely:

$$U_{\text{QN}} = \frac{\sum_i N_{\text{R}i}}{\frac{(n_{\text{QPU}}-1)M}{r}} = \frac{r\sum_i N_{\text{R}i}}{(n_{\text{QPU}}-1)M} \in [0,1], \qquad (2)$$

where r is the estimated execution time of a single non-local gate, $N_{\text{R}i}$ is number of non-local gates in the i-th job, $n_{\text{QPU}}-1$ is the maximum number of non-local gates in a layer that spans all the system's QPUs, and M is the makespan of the schedule. From Fig. 3, it is clear that the execution time of a non-local gate can be approximated with the entanglement generation time ($\approx 0.35\ s$), as it is at least three order of magnitude greater than any other operation involved. Also in this case, the shorter the makespan, the higher the quantum network utilization.

	U_{QPU}		U_{QN}	
Queue	FIFO	LS	FIFO	LS
1	0.668	0.856	0.465	0.597
2	0.748	0.859	0.521	0.599
3	0.856	0.859	0.597	0.599

TABLE IV: QPU and quantum network utilization resulting from different queues, scheduled with FIFO and LS.

The results provided in Table IV show that, in most cases (queue 1, queue 2), both QPU and Quantum network utilization are higher when LS is used. It may happen though that LS and FIFO are equivalent (queue 3). High QPU utilization is generally a good feature. High quantum network utilization could instead be an issue, as it may have a bad impact on the quality of the result. For this reason, it would better to have a job scheduling strategy that is aware of quantum network characteristics.

V. Conclusion

Distributed quantum computing is getting attention as a viable path towards large scale quantum computing. This work focused on execution management of DQC jobs, introducing a reasonable approach for estimating DQC job lengths. A few job scheduling examples were illustrated, using quantum circuits of practical interest, such as GHZ ones, and two very different algorithms (FIFO and List-Scheduling). By means of QPU and quantum network utilization metrics, it was possible to observe that, in most cases, both QPU and quantum network utilization are higher when List-Scheduling is used, rather than FIFO. Moreover, the paper shows that the QPU and quantum network utilization metrics proposed in a previous work for discrete parameters [6], remain valid even when more realistic continuous parameters are used.

Regarding future work, we plan to design novel job scheduling techniques that take into account also some specific quantum network features, such as entanglement rate and fidelity.

Acknowledgment

The authors acknowledge financial support from the EU Flagship on Quantum Technologies through the project Quantum Internet Alliance (EU Horizon Europe, grant agreement no. 101102140).

References

[1] M. Caleffi, M. Amoretti, D. Ferrari, J. Illiano, A. Manzalini, and A. S. Cacciapuoti, "Distributed Quantum Computing: a Survey," *Computer Networks*, 2024.

[2] D. Lago-Rivera, S. Grandi, J. V. Rakonjac, A. Seri, and H. de Riedmatten, "Telecom-heralded entanglement between multimode solid-state quantum memories," *Nature*, vol. 594, no. 7861, pp. 37–40, 2021.

[3] S. Wehner, D. Elkouss, and R. Hanson, "Quantum internet: A vision for the road ahead," *Science*, vol. 362, no. 6412, 2018.

[4] S. L. N. Hermans *et al.*, "Qubit teleportation between non-neighbouring nodes in a quantum network," *Nature*, vol. 605, no. 7911, pp. 663–668, 2022.

[5] J. V. Rakonjac, S. Grandi, S. Wengerowsky, D. Lago-Rivera, F. Appas, and H. de Riedmatten, "Transmission of light–matter entanglement over a metropolitan network," *Optica Quantum*, vol. 1, no. 2, pp. 94–102, Dec 2023.

[6] D. Ferrari and M. Amoretti, "A Design Framework for the Simulation of Distributed Quantum Computing," in *HPQCI Workshop, in conjunction with the 33rd ACM International Symposium on High-Performance Parallel and Distributed Computing*, 2024.

[7] R. Frantzeskakis, C. Liu, Z. Raissi, E. Barnes, and S. E. Economou, "Extracting perfect ghz states from imperfect weighted graph states via entanglement concentration," *Phys. Rev. Res.*, vol. 5, p. 023124, May 2023.

[8] D. Ferrari, S. Carretta, and M. Amoretti, "A modular quantum compilation framework for distributed quantum computing," *IEEE Transactions on Quantum Engineering*, vol. 4, pp. 1–13, 2023.

[9] M. Pompili, C. Delle Donne, I. te Raa, B. van der Vecht, M. Skrzypczyk, G. Ferreira, L. de Kluijver, A. J. Stolk, S. L. N. Hermans, P. Pawełczak, W. Kozlowski, R. Hanson, and S. Wehner, "Experimental demonstration of entanglement delivery using a quantum network stack," *npj Quantum Information*, vol. 8, no. 1, p. 121, Oct 2022.

[10] G. Avis, F. Ferreira da Silva, T. Coopmans, A. Dahlberg, H. Jirovská, D. Maier, J. Rabbie, A. Torres-Knoop, and S. Wehner, "Requirements for a processing-node quantum repeater on a real-world fiber grid," *npj Quantum Information*, vol. 9, no. 1, p. 100, Oct 2023.

[11] B. Johannes, "Scheduling parallel jobs to minimize the makespan," *Journal of Scheduling*, vol. 9, no. 5, pp. 433–452, 2006.

[12] R. A. Dutton, W. Mao, J. Chen, and W. Watson, "Parallel job scheduling with overhead: A benchmark study," in *2008 International Conference on Networking, Architecture, and Storage*, 2008, pp. 326–333.

[13] J. Sgall and G. J. Woeginger, "Multiprocessor jobs, preemptive schedules, and one-competitive online algorithms," *Operations Research Letters*, vol. 51, no. 6, pp. 583–590, 2023.

[14] D. Olliaro, M. Ajmone Marsan, S. Balsamo, and A. Marin, "The saturated multiserver job queuing model with two classes of jobs: Exact and approximate results," *SIGMETRICS Perform. Eval. Rev.*, vol. 51, no. 4, p. 28–29, feb 2024.

[15] J. de Jong, F. Hahn, N. Tcholtchev, M. Hauswirth, and A. Pappa, "Extracting GHZ states from linear cluster states," *Phys. Rev. Res.*, vol. 6, p. 013330, Mar 2024.