dqsweep: Parameter Sweeps for Benchmarking Distributed Quantum Computing Applications

Quang-Phong Tran[1,*], Claudio Cicconetti[1], Marco Conti[1] and Andrea Passarella[1]

[1]Consiglio Nazionale delle Ricerche – Istituto di Informatica e Telematica – 56124 Pisa, Italy

Abstract

Distributed quantum computing is a promising approach for deploying quantum technologies in High-Performance Computing infrastructures by pooling the resources of clusters of quantum processor units. However, the performance of an application in such a system depends on a large set of hardware and network parameters that are difficult to explore manually. Quantum network simulators provide a simulation environment, but they require users to write ad hoc scripts for each sweep and then post-process the data. We propose *dqsweep*, an open-source framework that automates this process and can provide new insights into the application's performance. A single command line runs exhaustive parameter sweeps, produces heat maps, and correlations ready to use. We have evaluated the framework on two distributed protocols for the nonlocal CNOT gate, implemented via a Two Teledata vs. Telegate approach, and a distributed Grover algorithm on two qubits and three qubits.

Keywords

Distributed Quantum Computing, Quantum Benchmarking, Quantum Network Simulation

1. Introduction

Big data applications require High-Performance Computing (HPC) infrastructures to tame the inherent size and complexity through the concurrent use of a vast number of computation/storage/network elements tightly interconnected. Quantum computing is an emerging technology with the potential to play a significant role in big data and HPC, but today still suffers from limitations in terms of noise and small scale. The latter can be overcome through Distributed Quantum Computing (DQC), where resources of clusters of Quantum Processor Units (QPUs) are pooled together [1]. These QPUs, which can be integrated into HPC facilities [2], can communicate and share information to combine their independent results and achieve a computation in a network that spans a range from short to possibly long distances. A recent experimental demonstration of DQC has shown promising results [3], notably the realization of the teleportation of a CZ gate with 86% fidelity and a distributed Grover on two qubits with 71% success rate. However, DQC encounters all the challenges associated with distributed classical systems, as well as new ones, such as the decoherence of qubits in memories, which sets a hard limit on the time available for a computation task, and the inherent fragility of entanglement distribution over a quantum network, which cannot rely on Forward Error Correction (FEC) and buffering due to the no-cloning theorem [4].

Given the technology's early stage, it is challenging to assess the performance of a distributed quantum application in a specific DQC system, as it combines multiple computing and networking parameters with possibly different hardware and network topologies. Quantifying the relative influence of one parameter on the system performance, compared to the others, is essential for the exploration of DQC capabilities. It can also help to design optimized distributed quantum applications, such as large-scale quantum circuits, and facilitate defining minimum requirements for each criterion to run

BigHPC2025: Special Track on Big Data and High-Performance Computing, co-located with the 4th Italian Conference on Big Data and Data Science, ITADATA2025, September 9 – 11, 2025, Torino, Italy.

*Corresponding author.

quang.tran@iit.cnr.it (Q. Tran); c.cicconetti@iit.cnr.it (C. Cicconetti); m.conti@iit.cnr.it (M. Conti); a.passarella@iit.cnr.it (A. Passarella)

0009-0002-1985-739X (Q. Tran); 0000-0003-4503-4223 (C. Cicconetti); 0000-0003-4097-4064 (M. Conti); 0000-0002-1694-612X (A. Passarella)

a distributed quantum application in a future real DQC system, thereby bridging the gap between theoretical and experimental values. However, there are no tools readily available for this crucial task.

To fill this gap, we propose a framework, called *dqsweep*, built on top of a widely adopted quantum network simulation tool (Netsquid/SquidASM), for the automated evaluation of the performance of distributed quantum applications. The main contributions are:

- Automatic generation of the combinations of specified parameters with a range of values, parallel execution of the simulations, and visualization of the results.
- Single command-line to sweep network and hardware parameters and reveal how they affect the system performance, in terms of fidelity and latency
- Use case validation: evaluation of the performance of two nonlocal CNOT gate protocols (Two Teledata vs. Telegate) and a distributed Grover on two qubits vs. three qubits with noisy hardware.

2. Background and related work

2.1. Distributed Quantum Computing

Quantum computing promises to solve some complex problems faster than a classical computer by leveraging quantum mechanics principles. However, in practice, a quantum computer is highly susceptible to errors due to decoherence and noise, which limits the size of the input and, consequently, the problems that can be solved on it. Quantum Error Correction (QEC) mechanisms exist to detect and correct errors [5], but implementing a "clean" logical qubit requires many noisy qubits, which are a scarce resource today. DQC is an alternative approach that aims to scale the number of qubits [6] and achieves fault-tolerant quantum computation. Clusters of QPUs can be interconnected on a single chip/board (similar to classical multi-core CPUs) [3, 7] or they might be part of a geographical network (quantum internet)[1, 8].

2.2. Nonlocal Gates

Some quantum operations, specifically nonlocal gates, may be involved in distributing an initial monolithic quantum circuit across multiple QPUs. These operations, specifically the multi-qubit gates, that could be engaged between multiple QPUs, are costly in terms of resources. Consequently, one of the main challenges of DQC is to propose an optimal implementation of nonlocal gates and minimize their use in distributed quantum applications. In quantum computing, the controlled-NOT gate (CNOT) is an essential two-qubit gate because all single-qubit gates with this gate have been demonstrated to be universal [9]. This means that implementing a nonlocal CNOT gate could (in theory) enable the implementation of any distributed quantum computation. The CNOT gate flips the target qubit, noted $|y\rangle$, if and only if the control qubit, noted $|x\rangle$, is in state $|1\rangle$.

One possible implementation of this gate uses two quantum teleportations (or two teledata). This method can be used to implement any two-qubit gate. It consumes two bits of classical communication in each direction and two shared ebits, as shown in Figure 1 (left). The need for two teleportations and a SWAP gate arises because after Alice teleports her data qubit, she no longer has this qubit after the measurement, so Bob needs to send back the initial qubit to her. Another possible implementation is to teleport the gate (telegate). One bit of classical communication in each direction and one shared ebit are necessary and sufficient to implement the nonlocal CNOT gate [10]. This implementation uses two primitive operations, the cat-entangler that takes an arbitrary control qubit $\alpha|0\rangle + \beta|1\rangle$ and a maximally entangled pair $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ and transforms them into a cat-like state $\alpha|00\rangle + \beta|11\rangle$ and the cat-disentangler, which is the inverse operation [11], illustrated in the circuit (Figure 1, right side).

2.3. Quantum Benchmarking

Quantum benchmarking is a reproducible evaluation method to assess the performance of a quantum setup. It is a well-studied field, and there are multiple types of quantum benchmarking [12]. Significant

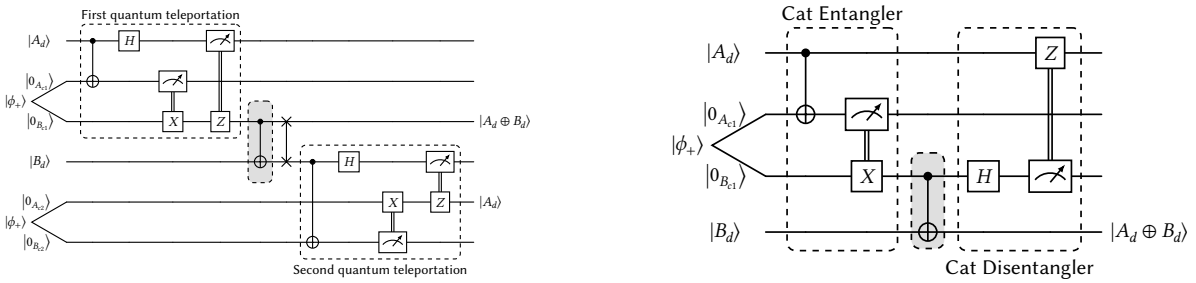

Figure 1: Comparison of two distributed CNOT circuits: two Teledata (left) vs. Telegate (right).

efforts have been proposed to support the development of quantum computing benchmarking at different levels, including the proposal of new frameworks [13, 14, 15] or metrics [16, 17]. Or by providing a suite of quantum circuits [18, 19, 20, 21]. However, the research community so far has focused only on monolithic quantum computing.

Similarly, significant effort has been devoted to developing quantum network simulators, such as Netsquid [22], SeQUeNCe [23], QuISP [24], and QuNetSim [25]. These tools allow the simulation of a quantum network at the hardware level (e.g., coherence time, gate execution times, noise) and facilitate the study of quantum protocols by providing a link layer (e.g, entanglement generation, network latency, photon loss). However, these simulators fall short of answering high-level development questions to design DQC applications: What will be the performance of an application, in the real system, if, for instance, the error rate of the CNOT gate goes through 1%, 3%, or 5%? Will the application still be helpful, and what will be the acceptable error rate? To answer this type of question, one must transform the hardware and network parameters into a "high-level metric" that enables the measurement of the success rate of an application, rather than the performance of a single, isolated hardware component. Then create an ad hoc script to sweep the parameters on a range of values, run the simulation manually a specified number of times, and present the statistical results with accompanying figures. This process is error-prone and time-consuming, depending on factors such as the number of applications, different network topologies, or parameter ranges.

There is a shortage of tools that combine the benchmarking of quantum computing and networking. To the best of our knowledge, the only tool has been provided by Liao et al. [26], who have performed a comparison study of a selection of quantum network protocols using NetSquid [22]. However, the software is designed for security-oriented applications and does not fit generic DQC, which has motivated us to develop *dqsweep*.

3. Methodology

3.1. *dqsweep* framework

dqsweep is a flexible framework that analyzes the performance of distributed quantum applications by sweeping quantum network and hardware parameters. The code is open-source and available on GitHub[1] (more DQC applications are provided in addition to the ones evaluated here). To simulate quantum networks, we used Netsquid (a network simulator using discrete events) [22] and SquidASM (an extension of Netsquid)[2]. *dqsweep* was written in Python 3.10 (but works with higher versions) and requires Netsquid credentials to be used. The quantum network can be represented as either a YAML file or an object of type `StackNetworkConfig`. It comprises multiple hardware parameters (T1, T2, gate execution times, noise model, ...) and network parameters (quantum nodes, quantum channel configurations, and classical channel configurations).

To evaluate these parameters given a distributed quantum application, the framework gives two

[1]https://github.com/Quang00/dqsweep
[2]https://github.com/QuTech-Delft/squidasm

primary performance indicators. Firstly, we use fidelity F, with $0 \leq F \leq 1$, a common quantitative measure, to quantify the accuracy of a quantum system by comparing two quantum states (e.g., final state vs. expected state), defined as density matrices. Typically, a fidelity of 1 means that the two quantum states are identical, while 0 means completely orthogonal. It is used to evaluate the usefulness and reliability of a DQC application. We define the **average fidelity** as follows:

$$\mu_F = \frac{1}{n} \sum_{i=1}^{n} \left(\mathrm{Tr}(\rho_i \sigma_i) + 2\sqrt{det(\rho_i) det(\sigma_i)} \right) \tag{1}$$

where n is the total number of simulations, ρ_i is the expected density matrix and σ_i the output density matrix for simulation i. Secondly, we measure latency, which is another critical indicator that has an important impact on the feasibility and performance of an application. In particular, due to decoherence, a high latency (e.g, classical communication latency, duration to successfully generate entanglement pairs, or gate times) could result in impracticality of a quantum protocol or poor outcomes. Therefore, we define the **average latency** as follows:

$$\mu_s = \frac{1}{n} \sum_{i=1}^{n} (t_{\text{end}}^{(i)} - t_{\text{start}}^{(i)}) \tag{2}$$

where $t_{\text{start}}^{(i)}$ and $t_{\text{end}}^{(i)}$ are the start and end times of simulation i, respectively.

A quantum network is associated with a quantum application to assess its performance. For each batch of experiments, *dqsweeep* performs the following operations:

1. Load the quantum network configuration (`--config`) and create the simulated quantum network.
2. Generate all combinations of values of the given parameters (`--sweep_params`) by sweeping specified ranges of values (`--ranges`).
3. Run the quantum distributed quantum application (`--experiment`) in parallel via multiprocessing (`ProcessPoolExecutor`) for each combination a certain amount of times (`--epr_rounds`, `--num_experiments`).
4. Store the results and generate three types of output:
   - A CSV file with the raw results that contains, for each row, the name of the parameters given in input associated with each swept value, and the list of fidelity/latency results, together with average and standard deviation.
   - Heat map plots, for all the performance indicators, for each value of the swept parameter.
   - A TXT file with the Pearson correlation for each parameter on the average fidelity and latency.

dqsweep is designed to be extensible and flexible. Users can define their quantum applications using modular classes in Python for Netsquid/SquidASM, with the condition that the results are fidelity and latency, as described above. The applications can be associated with a network topology by a YAML configuration or a Python object. The parameter sweeps accept any parameters with any corresponding valid range of values that are supported by the simulators and defined in the initial setup. This design makes *dqsweep* suitable for a wide range of research tasks involving the exploration of a large set of parameters in DQC.

3.2. Quantum Network and Noise Model

To evaluate our framework, we designed a quantum network with two quantum nodes, Alice (A) and Bob (B). Each quantum node has two qubits, one data qubit (denoted A_d for Alice and B_d for Bob) and one communication qubit (denoted A_c for Alice and B_c for Bob) where the Hilbert space $\mathscr{H}_i = \mathbb{C}_d^2 \otimes \mathbb{C}_c^2$ with $i \in \{A, B\}$. Each is a generic device that implements the following set of gates $\mathscr{G}$: Pauli gates (X, Y, Z), rotational gates (R_x, R_y, R_z), the Hadamard gate (H), controlled NOT gate ($CNOT$) and controlled Z gate (CZ).

For these devices, we used a noise model that gradually increases the probability of depolarization of single-qubit gates, noted $p_s \in \{0, ..., 0.05\}$, and the probability of depolarization of two-qubit gates, noted $p_t \in \{0, ..., 0.05\}$. The experiment starts with an initial pure state $|\psi_0\rangle$. The corresponding density matrix is defined as: $\rho_0 = |\psi_0\rangle\langle\psi_0|$ with $\rho_0 > 0$ and $\mathrm{tr}(\rho_0) = 1$. Suppose, for instance, that the system's evolution is described by the unitary operator $U \in \mathcal{G}$. After evolution, the initial state $|\psi_0\rangle$ will be in the state $U|\psi_0\rangle$. This evolution is described by $\rho_0 \xrightarrow{U} U\rho_0 U^\dagger$. Next, to describe the noise of the system, we will use the following notation for the single qubit depolarizing channel [27]:

$$\mathcal{E}_{p_s}(\rho) = (1 - p_s)\rho + \frac{p_s}{3}(X\rho X + Y\rho Y + Z\rho Z) \tag{3}$$

When $p_s = 0$, $\mathcal{E}_0 = \rho$, which is the identity channel, a noise-free quantum channel. For slight depolarizing noise, where $p_s \simeq 0$, the channel remains nearly identical to the identity channel. However, when the depolarizing noise increases, the qubit depolarizes and is entirely replaced by a maximally mixed state for $p_s = 1$ with $\mathcal{E}_1(\rho) = \frac{I}{2}$. Secondly, we defined the two-qubit depolarizing channel with $(i, j) \in \{0, 1, 2, 3\}^2 \setminus \{(0, 0)\}$ as:

$$\mathcal{E}_{p_t}(\rho) = (1 - \frac{15}{16}p_t)\rho + \frac{p_t}{16}\sum_{(i,j)}(\sigma_i \otimes \sigma_j)\rho(\sigma_i \otimes \sigma_j) \tag{4}$$

with $\sigma_0 = I$, $\sigma_1 = X$, $\sigma_2 = Y$ and $\sigma_3 = Z$.

3.3. Experimental Setup

We evaluate *dqsweep* on *MacBook Air (2020), Apple M1, 8GB*, to test the framework on four distributed quantum applications: a comparison of two protocols (Two Teledata vs. Telegate) that distribute a nonlocal CNOT gate, presented in section 2, and a performance analysis of distributed Grover on two qubits compared to its version on three qubits. We now briefly present the goal of Grover's algorithm [28], which is to find an element e in an unsorted database. In our evaluation, we assume that there is only one element to search for. We can formulate the search problem as follows: given a boolean function $f(x) : \{0, 1\}^n \to \{0, 1\}$, find e such that $f(x) = \begin{cases} 1 & \text{if } x = e \\ 0 & \text{else } x \neq e \end{cases}$, where $x \in \{0, 1\}^n$. To solve this problem classically, it requires querying the oracle in the worst case $\Theta(N)$ and on average $O(N)$ with N the total number of elements in the database. Instead, with Grover's algorithm, there is a quadratic gain, since the algorithm queries $O(\sqrt{N})$ times. The distributed version of Grover's algorithm implemented in *dqsweep* is a basic implementation that searches the state $e = |11\rangle$; it requires distributing two nonlocal controlled Z gates (CZ) on two qubits and two nonlocal CCZ gates on three qubits. The circuit on two qubits is represented in Figure 2.

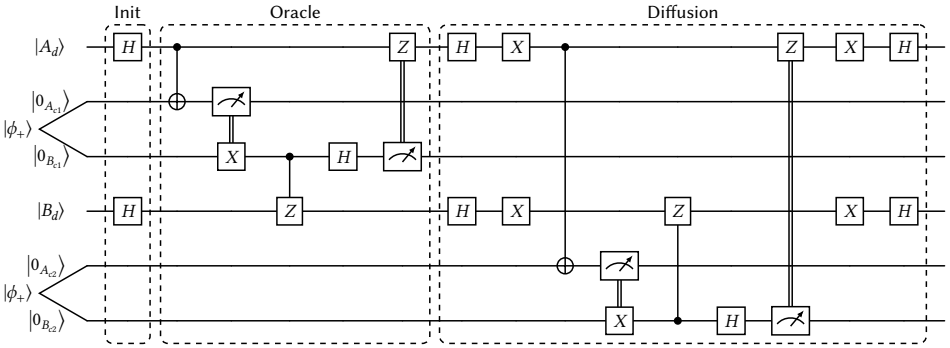

Figure 2: The distributed version of Grover, between two parties, Alice and Bob, to find the state $|11\rangle$. $|A_d\rangle, |B_d\rangle$ represents the state of the data qubit of Alice and Bob, respectively. $|A_c\rangle, |B_c\rangle$ represents the state of the communication qubit of Alice and Bob, respectively.

4. Results

We focus on two types of DQC applications: nonlocal CNOT gates (two protocols) and distributed Grover's algorithm (on two and three qubits). We study these protocols because distributed nonlocal gates are crucial for the realization of DQC. By benchmarking them, *dqsweep* can highlight protocol-level trade-offs. In addition, we evaluate a distributed Grover algorithm (real-world quantum application) that has been experimentally implemented [3] on two qubits; thus, by using *dqsweep* we can investigate the impacts of scalability and compare to the actual experimental values. The results below highlight how noise parameters affect fidelity and demonstrate the kinds of insights our framework can extract automatically.

The fidelity results are presented in Figures 3–6 in four heat maps and a correlation Table 1, where the y axis represents the single qubit gate depolarizing probability p_s and the x axis the two qubit gate depolarizing probability p_t. A square characterizes the result of the average fidelity $\mu_F = \frac{1}{n}\sum_{i=1}^{n} F(\rho_i, \sigma_i)$ from Eq. (1) with n the total number of simulations of the given (x_i, y_i) with $y_i = p_s$ and $x_i = p_t$. Along with each average fidelity, the sample standard deviation is provided, noted $s = \sqrt{\frac{\sum_i^N (x_i - \bar{x})^2}{N-1}}$, where x_i is the observed value of a sample, $\bar{x}$ is the mean value of observations, and N is the number of observations. In addition, for every pair of probabilities (p_s, p_t), which comprised a total of 225 combinations, the simulation was run 1,000 (100) times for distributed CNOT gates (distributed Grover), each consisting of 10 measurements.

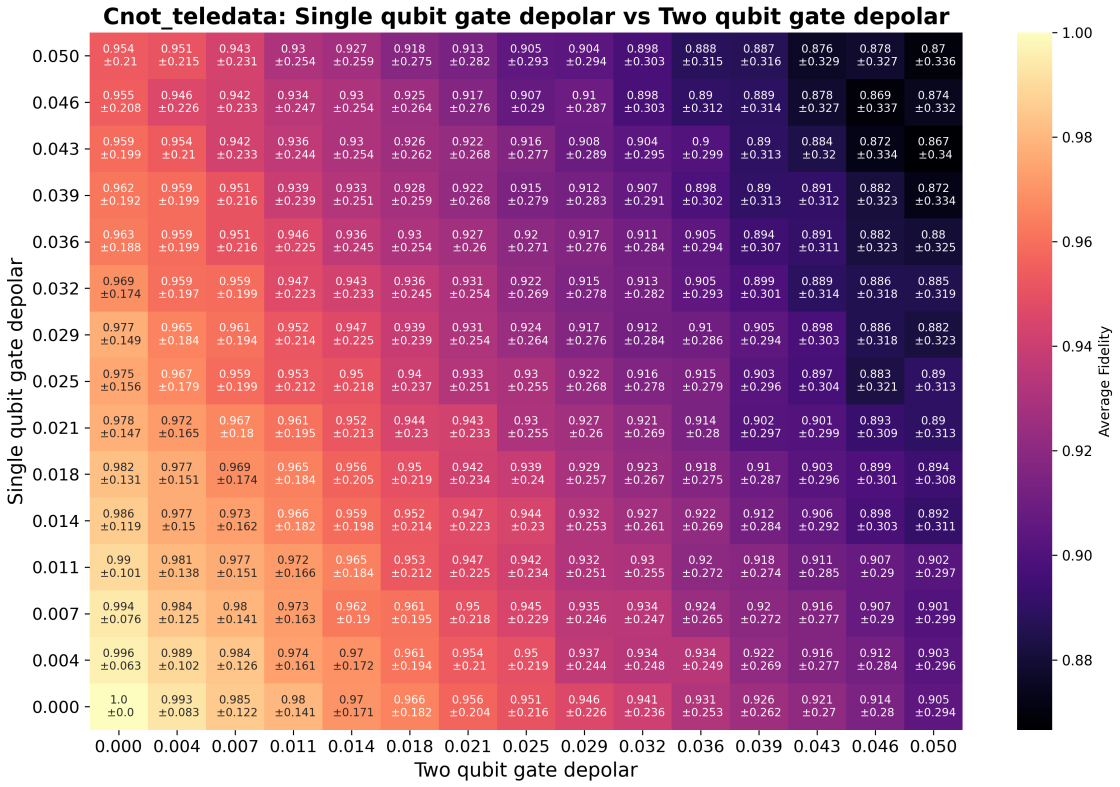

Figure 3: Heat maps of the average fidelity of the single qubit gate depolarizing probability p_s and the two qubit gate depolarizing probability p_t of distributed CNOT using two teledata.

Based on the heat maps, we observe that the maximum average fidelity, with the parameters $(p_s, p_t) = (0, 0)$, corresponding to no noise, is 1.0 for the CNOT two teledata, CNOT telegate, distributed Grover on two qubits, and 0.94 for distributed Grover on three qubits. The minimum average fidelity, with the parameters $(p_s, p_t) = (0.05, 0.05)$, corresponding to the "maximum" noise, is 0.87, 0.89, 0.57, 0.14 for the two teledata, telegate, Grover on two qubits, and Grover on three qubits, respectively.

Furthermore, *dqsweep* gives the Pearson Correlation Coefficient (PCC), noted ρ, defined as follows

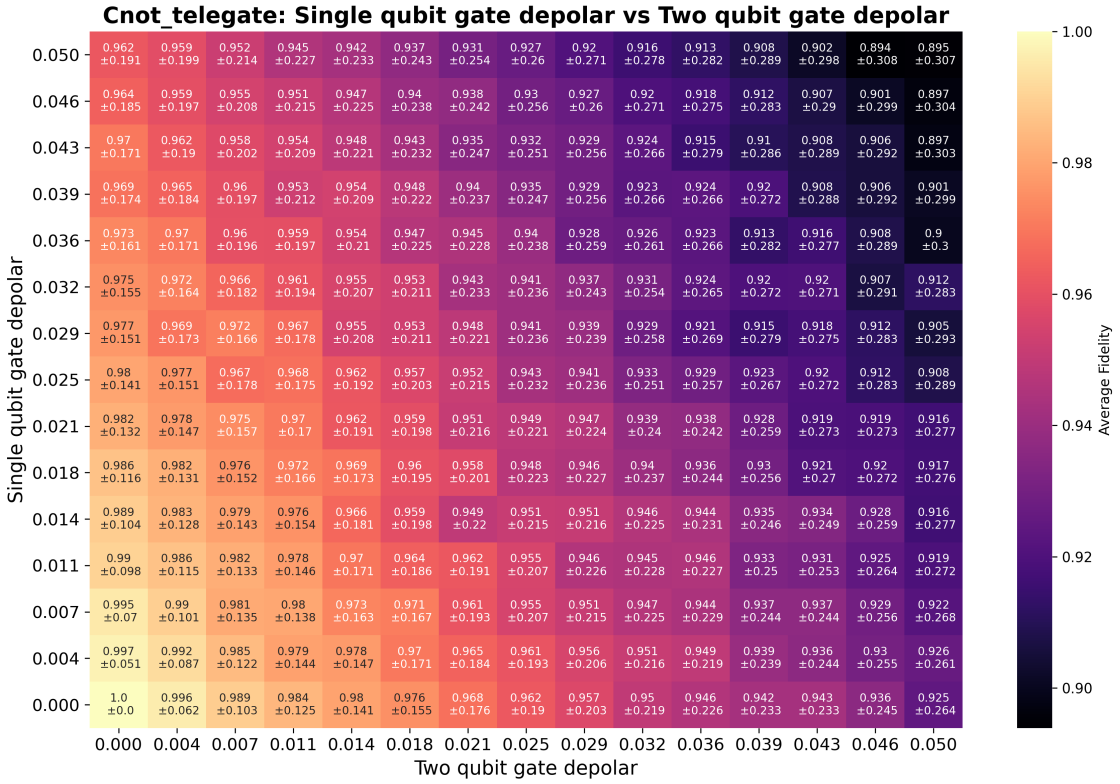

Figure 4: Same as Figure 3, but the distributed CNOT is implemented using telegate.

for the single qubit depolarizing probability p_s, $\rho_{p_s,\mu_f} = \frac{\text{cov}(p_s,\mu_f)}{\sigma_{p_s}\sigma_{\mu_f}}$ and conversely for the two qubit depolarizing probability p_t, $\rho_{p_t,\mu_f} = \frac{\text{cov}(p_t,\mu_f)}{\sigma_{p_t}\sigma_{\mu_f}}$ where cov is the covariance, μ_f is the average fidelity, σ_{p_s} is the standard deviation of p_s and σ_{p_t} is the standard deviation of p_t. The results are in the Table 1.

Table 1
PCC table between each parameter on the average fidelity of each distributed application

Parameters	Teledata	Telegate	Grover-2	Grover-3
Single qubit depolarizing probability (p_s)	-0.439	-0.452	-0.876	-0.559
Two qubit depolarizing probability (p_t)	-0.895	-0.887	-0.468	-0.565

5. Discussion and limitations

The results highlight several key aspects of the design and performance of distributed quantum applications. First, the correlation Table 1 shows that the two distributed protocols rely heavily on the two-qubit gates, and thus the degradation of the fidelity depends mainly on the noise of these gates. The telegate protocol exhibits slightly better fidelity than the teledata protocol, while also consuming fewer resources (only one ebit and two bits); therefore, this protocol appears to be the one preferred for distributing nonlocal gates. In contrast to the distributed gate protocols, the distributed Grover protocol on two qubits shows that single-qubit gate noise has a more significant impact on fidelity, while two-qubit gate noise has a moderate effect with the additional two nonlocal CZ gates. The distributed Grover on three qubits requires distributing two nonlocal CZZ gates, which involves four EPR pairs and several Toffoli gates (in the implementation, a composition of rotational gates with CNOT gates).

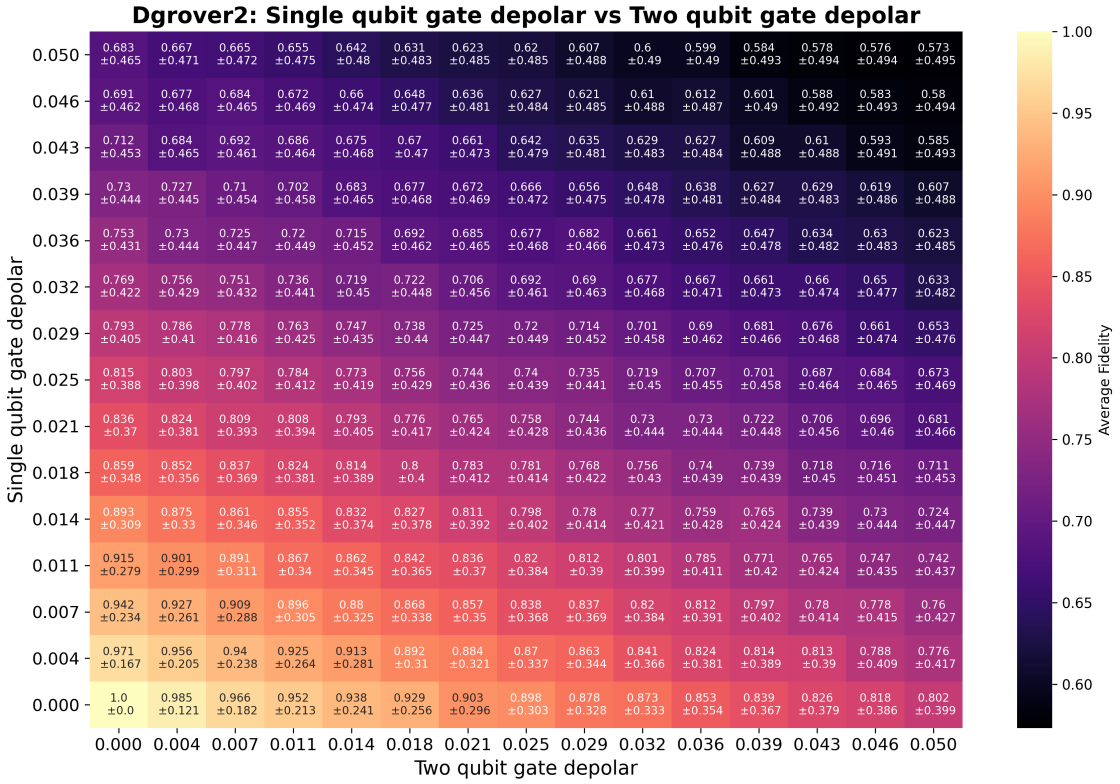

Figure 5: Same as Figure 3, but for the distributed Grover on two qubits, involving two nonlocal CZ gates.

By increasing the size by one qubit, the fidelity drops significantly to around 0.66 with $p_s \geq 0.004$ and $p_t \geq 0.004$, by a factor of 6 compared to Grover-2, which lost only around 5%. This is because the circuit depth is larger, and distributing a multi-qubit gate is costly. Each time a control qubit is added, an EPR pair needs to be generated, and more quantum operations are necessary to implement the multi-qubit gate.

Without *dqsweep*, exploring 225 combinations of noise parameters for each application would require extensive custom scripting and post-processing. The framework automates this process and enables direct visualization and statistical analysis. While *dqsweep* provides a new framework for benchmarking distributed quantum applications, the major limitation of our tool is that it relies on simulation, so there is still a need to experiment with real hardware to obtain accurate values for each parameter. Another possible improvement would be to integrate automatic circuit partitioning, a compiler that partitions a monolithic circuit across QPUs, instead of breaking it up manually. Finally, future work could develop a framework that provides the optimal configuration for a particular application with fixed parameters, or a specific given network, or specific requirements (e.g., the application needs a 95% fidelity; what is the minimal configuration?).

6. Conclusion

In this paper, we presented *dqsweep*, a flexible framework for performance analysis of a given distributed quantum application. By using Netsquid/SquidASM, *dqsweep*, in one command line, generates all the combinations of hardware and network parameters (specified in input), simulates the distributed quantum application in parallel for each combination, and produces heat maps, correlations, and a CSV raw results. This automation significantly reduces the experimental effort to analyze the impact of hundreds of possible configurations. It can also provide new perspectives (e.g., helpful windows) on application performance by comparing it at the same scale (range of parameters, range of values, heat maps) and the same statistical setup (number of EPR rounds, number of simulations). As a use

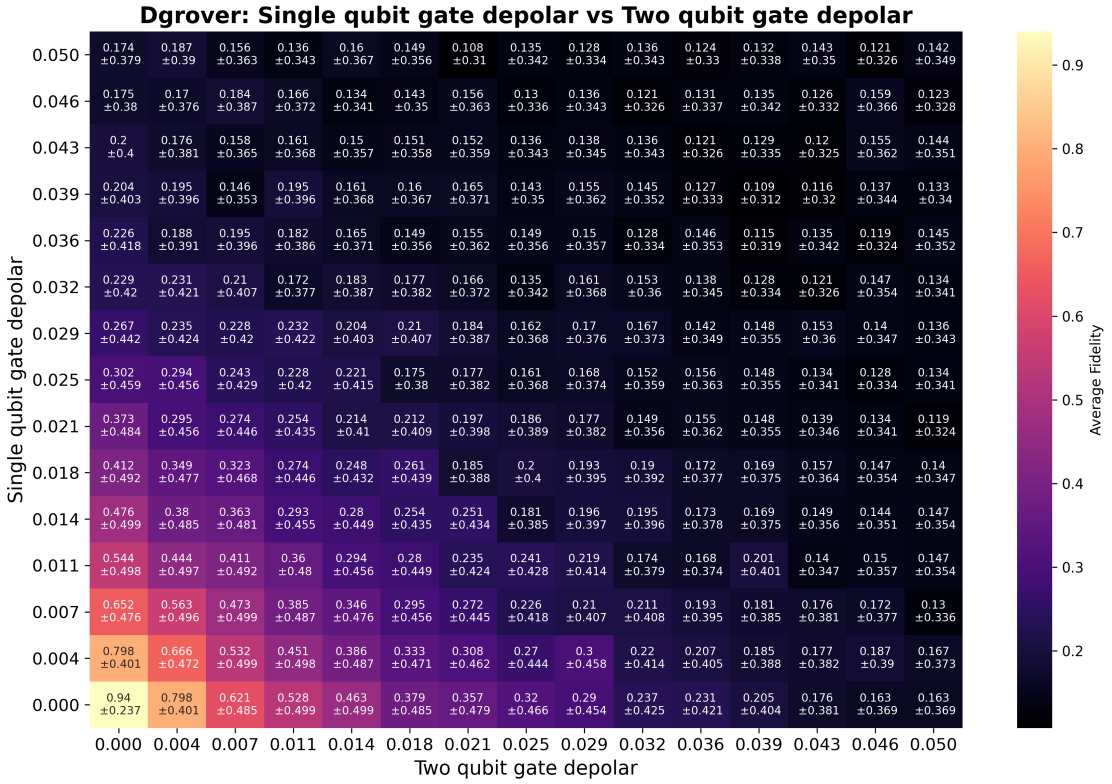

Figure 6: Same as Figure 3, but for the distributed Grover on three qubits, involving two nonlocal CCZ gates.

case, we have evaluated two distributed protocols of the nonlocal CNOT gate and the distributed Grover algorithm on two qubits versus three qubits. One notable result is that, when the depolarizing probabilities are 0.4%, the fidelity of the distributed Grover on three qubits drops by a factor of 6 compared to its version on two qubits.

Acknowledgments

The work of C. Cicconetti was supported by the European Union – Next Generation EU under the Italian National Recovery and Resilience Plan (NRRP), Mission 4, Component 2, Investment 1.3, CUP B93C22000620006, National Center for HPC, Big Data and Quantum Computing "ICSC"(CN00000013). The work of M. Conti and A. Passarella was funded by the European Union – Next Generation EU under the NRRP, Mission 4, Component 2, Investment 1.3, CUP B53C22003970001, partnership on "Telecommunications of the Future" (PE00000001 "RESTART", Spoke 1).

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] M. Caleffi, M. Amoretti, D. Ferrari, J. Illiano, A. Manzalini, A. S. Cacciapuoti, Distributed quantum computing: A survey, Computer Networks 254 (2024) 110672. doi:https://doi.org/10.1016/j.comnet.2024.110672.

[2] D. Barral, F. J. Cardama, G. Díaz-Camacho, D. Faílde, I. F. Llovo, M. Mussa-Juane, J. Vázquez-Pérez, J. Villasuso, C. Piñeiro, N. Costas, J. C. Pichel, T. F. Pena, A. Gómez, Review of distributed quantum

computing: From single qpu to high performance quantum computing, Computer Science Review 57 (2025) 100747. doi:https://doi.org/10.1016/j.cosrev.2025.100747.

[3] D. Main, P. Drmota, D. P. Nadlinger, E. M. Ainley, A. Agrawal, B. C. Nichol, R. Srinivas, G. Araneda, D. M. Lucas, Distributed quantum computing across an optical network link, Nature 638 (2025) 383–388.

[4] A. S. Cacciapuoti, M. Caleffi, F. Tafuri, F. S. Cataliotti, S. Gherardini, G. Bianchi, Quantum internet: Networking challenges in distributed quantum computing, IEEE Network 34 (2020) 137–143. doi:10.1109/MNET.001.1900092.

[5] P. Shor, Fault-tolerant quantum computation, in: Proceedings of 37th Conference on Foundations of Computer Science, 1996, pp. 56–65. doi:10.1109/SFCS.1996.548464.

[6] R. Van Meter, S. J. Devitt, The path to scalable distributed quantum computing, Computer 49 (2016) 31–42. doi:10.1109/MC.2016.291.

[7] A. Carrera Vazquez, C. Tornow, D. Ristè, S. Woerner, M. Takita, D. J. Egger, Combining quantum processors with real-time classical communication, Nature 636 (2024) 75–79.

[8] S. Wehner, D. Elkouss, R. Hanson, Quantum internet: A vision for the road ahead, Science 362 (2018) eaam9288.

[9] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, H. Weinfurter, Elementary gates for quantum computation, Phys. Rev. A 52 (1995) 3457–3467. doi:10.1103/PhysRevA.52.3457.

[10] J. Eisert, K. Jacobs, P. Papadopoulos, M. B. Plenio, Optimal local implementation of nonlocal quantum gates, Phys. Rev. A 62 (2000) 052317. doi:10.1103/PhysRevA.62.052317.

[11] A. Yimsiriwattana, Generalized ghz states and distributed quantum computing (2004). doi:10.1090/conm/381/07096.

[12] J. Eisert, D. Hangleiter, N. Walk, I. Roth, D. Markham, R. Parekh, U. Chabaud, E. Kashefi, Quantum certification and benchmarking, Nature Reviews Physics 2 (2020) 382–390.

[13] J. R. Finžgar, P. Ross, L. Hölscher, J. Klepsch, A. Luckow, Quark: A framework for quantum computing application benchmarking, in: 2022 IEEE International Conference on Quantum Computing and Engineering (QCE), 2022, pp. 226–237. doi:10.1109/QCE53715.2022.00042.

[14] S. Machnes, U. Sander, S. J. Glaser, P. de Fouquières, A. Gruslys, S. Schirmer, T. Schulte-Herbrüggen, Comparing, optimizing, and benchmarking quantum-control algorithms in a unifying programming framework, Phys. Rev. A 84 (2011) 022305. doi:10.1103/PhysRevA.84.022305.

[15] R. Blume-Kohout, K. C. Young, A volumetric framework for quantum computer benchmarks, Quantum 4 (2020) 362. doi:10.22331/q-2020-11-15-362.

[16] A. W. Cross, L. S. Bishop, S. Sheldon, P. D. Nation, J. M. Gambetta, Validating quantum computers using randomized model circuits, Phys. Rev. A 100 (2019) 032328. doi:10.1103/PhysRevA.100.032328.

[17] S. Martiel, T. Ayral, C. Allouche, Benchmarking quantum coprocessors in an application-centric, hardware-agnostic, and scalable way, IEEE Transactions on Quantum Engineering 2 (2021) 1–11. doi:10.1109/TQE.2021.3090207.

[18] N. Quetschlich, L. Burgholzer, R. Wille, MQT Bench: Benchmarking Software and Design Automation Tools for Quantum Computing, Quantum 7 (2023) 1062. doi:10.22331/q-2023-07-20-1062.

[19] T. Tomesh, P. Gokhale, V. Omole, G. S. Ravi, K. N. Smith, J. Viszlai, X.-C. Wu, N. Hardavellas, M. R. Martonosi, F. T. Chong, Supermarq: A scalable quantum benchmark suite, in: 2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA), 2022, pp. 587–603. doi:10.1109/HPCA53966.2022.00050.

[20] A. Li, S. Stein, S. Krishnamoorthy, J. Ang, Qasmbench: A low-level quantum benchmark suite for nisq evaluation and simulation, ACM Transactions on Quantum Computing 4 (2023). doi:10.1145/3550488.

[21] T. Lubinski, S. Johri, P. Varosy, J. Coleman, L. Zhao, J. Necaise, C. H. Baldwin, K. Mayer, T. Proctor, Application-oriented performance benchmarks for quantum computing, IEEE Transactions on Quantum Engineering 4 (2023) 1–32. doi:10.1109/TQE.2023.3253761.

[22] T. Coopmans, R. Knegjens, A. Dahlberg, D. Maier, L. Nijsten, J. De Oliveira Filho, M. Papendrecht,

J. Rabbie, F. Rozpędek, M. Skrzypczyk, L. Wubben, W. De Jong, D. Podareanu, A. Torres-Knoop, D. Elkouss, S. Wehner, NetSquid, a NETwork Simulator for QUantum Information using Discrete events, Communications Physics 4 (2021) 164. doi:10.1038/s42005-021-00647-8.

[23] X. Wu, A. Kolar, J. Chung, D. Jin, T. Zhong, R. Kettimuthu, M. Suchara, Sequence: a customizable discrete-event simulator of quantum networks, Quantum Science and Technology 6 (2021). doi:10.1088/2058-9565/ac22f6.

[24] R. Satoh, M. Hajdusek, N. Benchasattabuse, S. Nagayama, K. Teramoto, T. Matsuo, S. A. Metwalli, P. Pathumsoot, T. Satoh, S. Suzuki, R. V. Meter, Quisp: a quantum internet simulation package, in: 2022 IEEE International Conference on Quantum Computing and Engineering (QCE), IEEE, 2022, p. 353–364. doi:10.1109/qce53715.2022.00056.

[25] S. DiAdamo, J. Nötzel, B. Zanger, M. M. Beşe, Qunetsim: A software framework for quantum networks, IEEE Transactions on Quantum Engineering (2021). doi:10.1109/TQE.2021.3092395.

[26] C.-T. Liao, S. Bahrani, F. F. da Silva, E. Kashefi, Benchmarking of quantum protocols, Scientific Reports 12 (2022) 5298. doi:10.1038/s41598-022-08901-x.

[27] M. A. Nielsen, I. L. Chuang, Quantum Computation and Quantum Information: 10th Anniversary Edition, Cambridge University Press, 2010.

[28] L. K. Grover, A fast quantum mechanical algorithm for database search, 1996. arXiv:quant-ph/9605043.