Not All Qubits Are Created Equal

A Case for Variability-Aware Policies for NISQ-Era Quantum Computers

Swamit S. Tannu
swamit@gatech.edu
Georgia Institute of Technology
Atlanta, Georgia

Moinuddin K. Qureshi
moin@gatech.edu
Georgia Institute of Technology
Atlanta, Georgia

Abstract

Existing and near-term quantum computers may not be large enough to support fault-tolerance. Such systems with few tens to few hundreds of qubits are termed as *Noisy Intermediate Scale Quantum computers (NISQ)*, and these systems can provide benefits for a class of quantum algorithms. In this paper, we study the problem of *Qubit-Allocation* (mapping of program qubits to machine qubits) and *Qubit-Movement* (routing qubits from one location to another for entanglement) for NISQ machines.

We observe that there can be variation in the error rates of different qubits and links, which can impact the decisions for qubit movement and qubit allocation. We analyze publicly available characterization data for the IBM-Q20 to quantify the variation and show that there is indeed significant variability in the error rates of the qubits and the links connecting them. We show that the device variability has a significant impact on the overall system reliability. To exploit the variability in error rate, we propose *Variation-Aware Qubit Movement (VQM)* and *Variation-Aware Qubit Allocation (VQA)*, policies that optimize the movement and allocation of qubits to avoid the weaker qubits and links, and guide more operations towards the stronger qubits and links. Our evaluations, with a simulation-based model of IBM-Q20, show that Variation-Aware policies can improve the system reliability by up to 1.7x. We also evaluate our policies on the IBM-Q5 machine and demonstrate that our proposal significantly improves the reliability of real systems (up to 1.9X).

CCS Concepts • **Hardware → Quantum technologies**.

Keywords Compilers, Quantum Computers, Noisy Intermediate Quantum Computers

ASPLOS '19, April 13–17, 2019, Providence, RI, USA

ACM ISBN 978-1-4503-6240-5/19/04…$15.00
https://doi.org/10.1145/3297858.3304007

ACM Reference Format:
Swamit S. Tannu and Moinuddin K. Qureshi. 2019. "Not All Qubits Are Created Equal: A Case for Variability-Aware Policies for NISQ-Era Quantum Computers" . In *2019 Architectural Support for Programming Languages and Operating Systems (ASPLOS '19), April 13–17, 2019, Providence, RI, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3297858.3304007

1 Introduction

Quantum computers can accelerate conventionally hard problems such as prime-factorization, and simulation of materials and molecules [10, 27]. Quantum algorithms use quantum bits (qubits) to exploit the properties of superposition and entanglement and rely on quantum operations to change the state of the qubits. In the last two decades, the field of quantum computing has moved from theoretical ideas to realizable systems (albeit at a small scale). The last two years represent significant milestones in the field of quantum computing, as Google IBM, and Intel have announced blueprints for quantum computers with 72, 50 and 49 qubits respectively [11, 15, 16]. Theses prototype quantum computers provide an opportunity to understand the challenges in building a practical quantum computer and use these insights to improve the design of future quantum computers.

Qubits are fickle as the qubit devices can lose state due to decoherence or operational errors. Qubits can be protected against errors using *Quantum error correction codes (QEC)*. Unfortunately, QEC requires significant overheads, typically incurring 10-100 physical qubits to encode one fault-tolerant qubit. Existing and near-term quantum computers with tens to hundreds of qubits may not have the capacity to utilize QEC due to the limited number of qubits. Such quantum computers with 10 to 1000 noisy qubits are termed as *Noisy Intermediate Scale Quantum computers (NISQ)* [25]. Even though NISQ machines lack fault-tolerance, they can still provide benefits for a class of quantum applications [22]. In this paper, we study policies for Qubit-Movement (routing a data-qubit from one location to another) and Qubit-Allocation (mapping of program qubits to the physical qubits) for NISQ machines.

Quantum computer harness its power from the ability to create an entangled collective state. An entangled state is generated by coupling a pair of qubits using two-qubit operation. A quantum machine can entangle only the qubits that have a link between them. Existing quantum computers

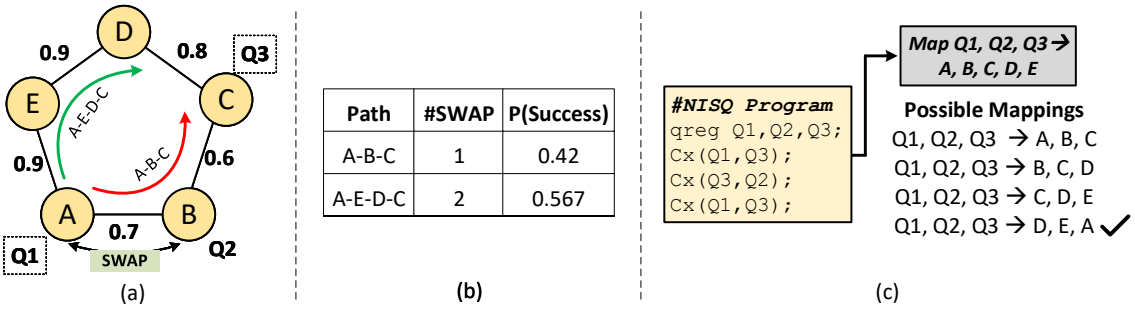


Figure 1. (a) A hypothetical quantum computer with five-qubits – the number on the edge denotes the success probability when that edge is used (b) Variation-Aware Qubit Mapping (VQM) can use more SWAP instructions and yet have higher probability of success (c) Variation-Aware Qubit Allocation (VQA) select the mapping that improves overall system reliability.

from IBM, Google, and Intel are designed using networks that offer limited connectivity, only to a few of the neighboring qubits, and this connectivity dictates the qubits that can be entangled. For example, Figure 1(a) shows a hypothetical quantum computer with five qubits where circular nodes represent the qubits and edges represent the coupling links between qubits. A pair of qubits can only be entangled if there exists a coupling link between them. Fortunately, quantum computers provide a *SWAP* instruction that can exchange the state of two neighboring qubits. For example, we want to entangle data qubit Q_1 and data qubit Q_3 which are initially residing at physical qubit-A, and physical qubit-C respectively. We can perform this operation in two steps: first swap the data between qubit-A and qubit-B such that Q_1 and Q_2 interchanges positions. Next, entangle qubit data Q_1 and Q_3. In quantum programs, a large number of SWAP instructions are inserted to move data so that entanglement between arbitrary qubits can be performed.

The *Qubit-Movement* policy deals with the problem of selecting a route to move the state of one qubit to another. For example, in Figure 1(a), we may choose the route A-B-C for going from A to C, as doing so would minimize the number of SWAP operations. The *Qubit-Allocation* policy deals with the problem of mapping of program qubits to the physical qubits. For example, in Figure 1(a), if we want to map three program qubits to five physical qubits, we would choose any of 3 connected qubits (for example, Q_1 maps to A, Q_2 maps to B, and Q_3 maps to C), as placing qubits nearby results in efficient movement. Prior studies [26, 28, 34] have proposed qubit allocation policies based on minimizing the number of SWAPs. These studies assume uniformity in the cost of performing SWAPs. However, in reality, we expect variation in the behavior of different qubits and links, and optimizing for a uniform behavior may not result in the best policy when device variation is taken into account.

To demonstrate that there is variation in the error-rates of different qubits and links, we analyze the *publicly-available* characterization data for the IBM-Q20 (20 qubits) machine. Such a characterization is performed for the IBM-Q20 several times a day, and we analyze the data for 52 days. We present the statistics of coherence time for all the 20 qubits, the error rate in performing single-qubit operations, and the error-rate in performing two-qubit operations across different qubits. For all these metrics we observe significant variation in the behavior of different qubits and links – in essence, qubits and links are not created equal. For example, our detailed analysis for the links connecting different qubits show that the error rates can vary by as much as 7x across different links in the system. Such variation can have a significant impact on the overall system reliability (Section 3).

To analyze the impact of variation on the overall system reliability, we use the *Probability of Successful Trial (PST)* metric. The PST metric indicates the probability that the program finished successfully without any error. As IBM-Q20 is not open to the public, we build a reliability evaluation infrastructure to compute the PST for the IBM-Q20 machine using the machine configuration and error rates based on the characterization data. Our evaluations show that the device variation has a significant impact on the system reliability. To improve system reliability, we should steer more instructions and movement to strong qubits and links, and fewer instructions and movement on weaker qubits and links. We propose such *Variation-Aware* policies to exploit the variation in the behavior of qubits and links, assuming error-rates are known at compile time (Section 4).

We propose *Variation-Aware Qubit Movement (VQM)* policy that routes the qubit from source to destination based on minimizing the probability of failure. For example, in Figure 1, the success probability of each link is denoted as a weight of the edge. Let us assume, we want to entangle data qubit Q_1 and data qubit Q_3. A conventional variation-unaware policy will use a path that minimizes the number of SWAP instructions, taking the path A-B-C, resulting in an overall probability of success of 42% for these operations. With VQM, we would take the route A-E-D-C, even though this route has more SWAP instructions, since it has an overall probability of success of 56.7%, as shown in Figure 1(b). VQM shows a significant improvement in PST (Section 5).

We also propose *Variation-Aware Qubit Allocation (VQA)* policy that performs the mapping of program-qubit to physical-qubit with the aim of improving the overall system reliability. For example, in Figure 1(c), we want to allocate three program qubits to 5 physical qubits. A conventional mapping policy can choose any of the listed mapping possibilities as they all would have similar cost in terms of SWAP operations. However, with VQA, we would use the mapping D, E, A, as this mapping uses the strongest links, and would improve the overall system reliability. We extend prior proposals for Qubit-Allocation with VQA and show that VQA+VQM can improve the PST of the IBM-Q20 by up to 1.7x (Section 6).

In addition to the simulation-based studies for IBM-Q20, we evaluate our proposed policies on a real quantum machine (IBM-Q5) and demonstrate that our policies continue to provide a significant improvement on the system reliability even in a realistic setting. VQM+VQA improves the PST of the IBM-Q5 by up to 1.9x (and average 1.36x) (Section 7).

We also perform a case study, where we analyze programs that require less than half the available qubits, and we have an option of either executing two copies of the program concurrently (to increase the rate of trials) or executing only one copy but map the work on strongest qubits and links (to improve the PST of the given trial). We demonstrate that, in certain cases, having one strong copy has better overall performance (successful trials per unit time) than having two concurrently running copies. Thus, variation-awareness can enable intelligent partitioning for NISQ machines (Section 8).

2 Background and Motivation

In this section, we provide a brief background on quantum computing, discuss the issues of errors and error correction, present a usage model for NISQ computers, and then discuss the problems associated with the NISQ machines.

2.1 Background on Quantum Computing

Conventional computers use binary data representation. In contrast, a quantum computer represents data using quantum bits (qubits). Consider a sphere, where the binary data can either be at the north pole or the south pole of the sphere, and conventional digital computers operate by switching the data between the two poles. In quantum computing, the state of a qubit can be viewed as an arbitrary point on the sphere that is a superposition of two basis states as shown in Figure 2(a). Quantum operations manipulate the state of the qubit by moving it from one point to another point on the sphere, as shown in Figure 2(b) and Figure 2(c). The ability to store and manipulate the state of qubits is key to quantum algorithms. The second property that facilitates quantum speedup is *entanglement*. Entanglement is the ability to produce a collective state of multiple qubits that are correlated. The entangled states are produced using two-qubit operations such as the *Controlled-NOT (CNOT)* instructions [21].

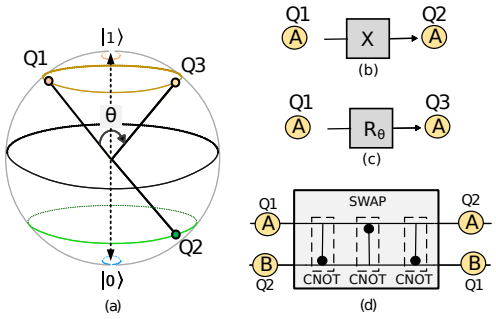


Figure 2. (a) Bloch Sphere representation of qubit. (b)–(c) Quantum operations manipulate the state by moving the point on sphere. (d) SWAP instruction interchanges the data of two qubits and can be done using 3 CNOT operations.

On IBM quantum machines, two-qubit operations are performed using a coupling-link that connects two qubits. For practical reasons, superconducting quantum computers do not allow all-to-all connectivity between the qubits and use a restricted network (such as Mesh) that allows connectivity between only the neighboring qubits. The network structures impose constraints on which qubits can be entangled. Fortunately, there are SWAP operations that can move the qubit from one location to another and enables entanglement of any two arbitrary qubits. Even if the quantum machine does not provide a native SWAP instruction, it can be accomplished using 3 CNOT gates, as shown in Figure 2(d).

2.2 Errors on Quantum Computers

Qubits are fickle as even a small perturbation in the environment can change the state of a qubit. The error rate for a qubit can be defined as the probability of undesired change in the qubit state. Errors in quantum computers can be classified into two categories: retention-errors or operational-errors.

Retention Errors (or Coherence Errors): A qubit can retain data for only a limited time, and this duration is called as *Coherence Time*. There are two types of retention errors that can occur, and there are two metrics to specify the coherence time of a quantum device. A qubit in a high-energy state ($|1\rangle$) naturally decays to the low-energy state ($|0\rangle$), and the time constant associated with this decay is called as the *T1 Coherence Time*. T1 indicates the time for natural relaxation of a qubit. However, there is also a possibility that qubit might interact with the environment and encounter a phase error, and the time constant associated with this decay is called the *T2 Coherence Time*. T2 indicates the time for a qubit to get affected by the environment.

The coherence times for superconducting quantum computers have improved from 1 nanosecond to 100 microseconds in the last decade [6]. Furthermore, existing superconducting qubits show improving trend in coherence times [15][6].

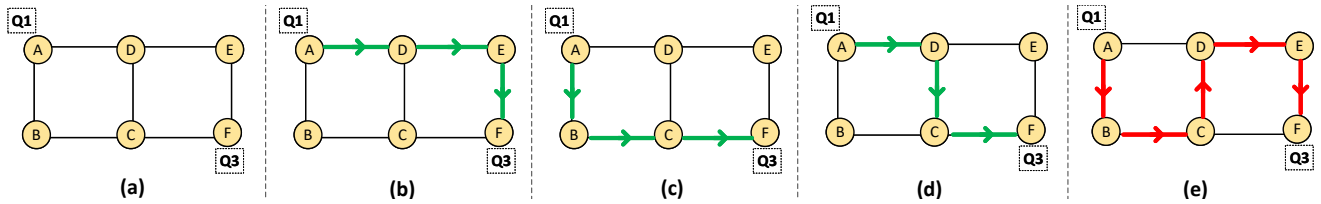


Figure 3. (a) Layout of a 6-qubit quantum computer, (b)-(e) are possible routes from A to F. Note that options (b)(c)(d) have an identical number of swaps and (e) incur higher swaps. An intelligent policy would choose one from (b)(c)(d).

Operational Errors (or Gate Errors): Performing operations on qubits can also affect their state incorrectly due to errors, as quantum operations are not perfect. For example, an instruction that rotates the state by some desired angle can introduce extra erroneous rotation. Operational error-rate is defined as the probability of introducing an error while performing the operation [17]. For publicly available quantum-computers from IBM, the single-qubit instruction error-rates are of the order of 10^{-3}, whereas for two-qubit instructions, such as CNOT, it is 10^{-2}. A typical quantum program contains a significant number of two-qubit operations, and given the error-rate of two-qubit operations are an order of magnitude higher than for the single-qubit operations, the two-qubit operations usually dominate the overall error rate. In this paper, we focus on operational errors, and specifically the ones caused by two-qubit operations.

2.3 Near-term Quantum Computers

Quantum computers can be made resilient to errors by using *Quantum Error Correction (QEC)* codes. Unfortunately, QEC requires a large number of physical qubits (10x-100x) to encode one fault-tolerant bit. This 10x-100x overhead in terms of physical qubits for performing error correction may be acceptable when the quantum machines have thousands of qubits. However, the current and near term quantum machines will not have enough capacity to implement QEC.

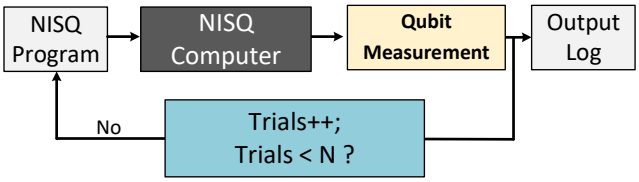


Figure 4. Iterative Computing Models for NISQ

Executing large-scale quantum application, such as Shor's factoring algorithm, requires having a quantum computer with millions of qubits. Existing quantum technologies are not mature enough to have millions of qubits. In fact, for existing quantum computers (fifty-plus qubits) or near-term quantum computers (with few hundreds of qubits), it may be impractical to perform error correction even for an application requiring few dozens of logical qubits. However, there exists a class of applications highlighted by Preskill [25] that can still be viable with such *Noisy and Intermediate-Scale Quantum (NISQ) computing*. Even though NISQ machines may not have enough resources for error correction, they rely on application properties to perform useful work. Figure 4 describe a general computing model for the NISQ machines. In this model, the given program is run multiple times and the log of the output is stored after each trial. As long as the correct results appear with non-negligible probability, we can infer the correct results by analyzing the output log.

2.4 Restricted Connectivity Between Qubits

In this paper, we focus on the problems due to the architecture of NISQ computers. If a NISQ computer contains N qubits, then ideally all the qubits will be connected to all other qubits. Such unrestricted connectivity would allow any two arbitrary qubits to get entangled. Unfortunately, such an organization would require $O(N^2)$ links, which is impractical even for the 49-72 qubits machines that are available today. The links in a quantum machine are not just wires, but resonators that operate at a dedicated frequency, and having a large number of such circuits operate reliably on the chip is a difficult task. Therefore, almost all qubit machines use a Mesh network (or a variant that allows diagonal connections). Such networks restrict that the movement of qubits can occur only between neighboring qubits. For example, for the hypothetical 6-qubit machine shown in Figure 3(a) there is no direct connection between qubits A and F. The communication between these qubits must happen via intermediate qubits. Such restrictions give rise to the two sub-problems: (a) Qubit-Movement policy, and (b) Qubit-Allocation policy.

Qubit-Movement Policy: This policy decides the route that should be used while moving the data from one location on the chip to another. Given that such movement is done using SWAP instructions between neighboring qubits, it is reasonable to select the route that minimizes the number of SWAP instructions. Figure 3(b)-(e) shows the four possible routes from A to F. The first three (b)-(d) requires only 3 SWAP operations, while (e) requires 4 SWAP operations. The policy may arbitrarily pick one of the routes from (b)-(d).

Qubit-Allocation Policy: This policy decides the initial mapping of program qubits to the data qubits. For example, it is preferred that qubits that communicate frequently be placed near each other. For example, if we wanted to place 4 qubits on the machine shown in Figure 3(a), we would not keep these qubits on the four corners, and instead we will try to use the middle two qubits (D and E), as doing so would minimize the SWAPs, required for communication. In fact, recent studies [26, 28, 34] have proposed such allocation policies based on minimizing the number of SWAPs.

In this paper, we use the compiler developed by Zulehner et al. [34] as the baseline for both Qubit-Allocation and Qubit-Movement. Our baseline compiler compiles the quantum program for given connectivity to generate the instruction schedule (with additional extra swaps) and initial program-qubit to physical-qubit mapping. It is designed to minimize the number SWAPs by using a greedy search algorithm. Baseline policy for Qubit-Movement and Qubit-Allocation assume uniform cost (specifically reliability impact) in performing SWAP operations. However, in reality, there can be significant variation in reliability of qubits and the links. Policies that take this variation into account can provide better overall system behavior (performance, reliability etc.) To enable such variation-aware policies, we first analyze the publicly available characterization data for the IBM-Q20 machine as IBM-Q20 has the most number of qubits for which characterization data is publicly available.

3 Analyzing Variation in IBM-Q20

To understand and quantify the variation in the error-rates of different qubits and links, we analyze the *publicly-available* characterization data for the IBM-Q20 (20-qubit) machine. IBM provides the data for the link error rates and the coherence times by publishing it on the IBM quantum experience web-page [5]. We monitored the IBM website for 52 days and gathered more than 100 different characterization reports. The characterization reports consist of error-rate for all single-qubit operations, two-qubit operations (link errors), and measurement operations. IBM machines are calibrated (one or more times) every day and error-reports are updated after each calibration cycle.

3.1 Distribution of Coherence Times

Both T1 and T2 coherence time of a qubit depends on several design, manufacturing and experimental parameters. Due to process variation, biasing and temperature drifts the coherence time can vary significantly. Figure 5 shows the T1 and T2 distribution of IBM-Q20. The data is collected for all 20 qubits over 100 observations (so a total of 2000 data points are plotted in the graph). The mean and standard deviation for T1-Coherence time are 80.32μS and 35.23μS respectively. The mean and standard deviation for T2-Coherence time are 42.13μS and 13.34μS respectively.

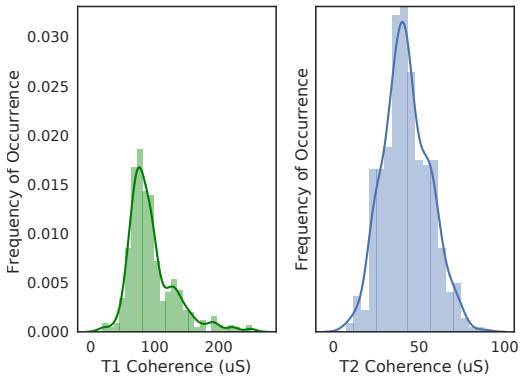


Figure 5. Distribution of (a) T1 Coherence time (b) T2 Coherence time for all 20 qubits with 100 samples per qubit

3.2 Error-Rate for Single-Qubit Operations

Single qubit operations rotate the quantum state from one point to other on a state-sphere. On IBM machine, it is performed by applying a microwave signal with a set duration and frequency on the qubit device. Unfortunately, qubit devices are highly non-linear and a small perturbation or experimental conditions can cause drift in device characteristics. This can cause variation in the robustness of the quantum operations. Figure 6 shows the distribution of error-rate for single-qubit operations. The data shows a large fraction of the error-rate below 1%. Single-qubit operations are more robust than two-qubit operations.

Figure 6. Distribution of the error-rates of single-qubit operation for all 20 qubits with 100 samples per qubit.

3.3 Error-Rate of Two-Qubit Operations

Two-qubit operations are essential to entangle quantum states and move the state of the qubits. In IBM quantum computers, two-qubit operations are performed by applying microwave pulses on target devices, control qubit devices as well as on the coupling link that connects the two. Similar to single-qubit operations, two-qubit operations suffer from variation in error-rate i.e. there is a fraction of coupling links significantly unreliable than most of the links. We analyze the reliability of two-qubit operations for the IBM quantum computer. Figure 7 shows the distribution of the error-rate

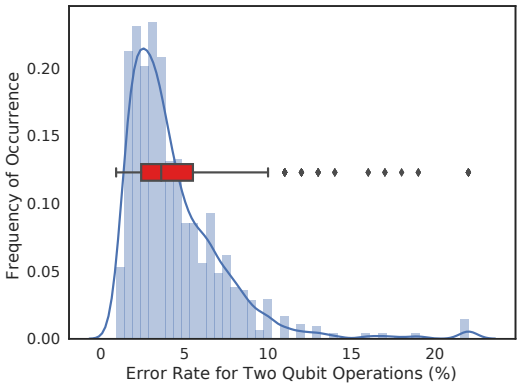


Figure 7. Distribution of the error-rates of two-qubit operations for all 76 links of IBM-Q20. The data consists of 100 observations for per link (so a total of 7600 datapoints).

of two-qubit operations for the 20 qubit machine. It consists of data from 76 coupling links collected over 100 calibration cycles. The mean two-qubit error-rate is 4.3% and standard-deviation is 3.02%.

3.4 Temporal Variation in Two-Qubit Gate Errors

Error-rate of a link can change with time. IBMQ-20 are frequently re-calibrated to ensure that the characterization is reliable. However, a qubit and the associated coupling links can change their behavior across two different calibration cycles. For example, a qubit pair with a low error rate on one day can have opposite behavior on the other. This might result from tuning parameters, drifts, and other experimental factors. Figure 8 shows a time-series of error-rate for three coupling-links. From this data, we observe that error-rate of the links tend to retain their mean error characteristics and stronger links tend to remain strong.

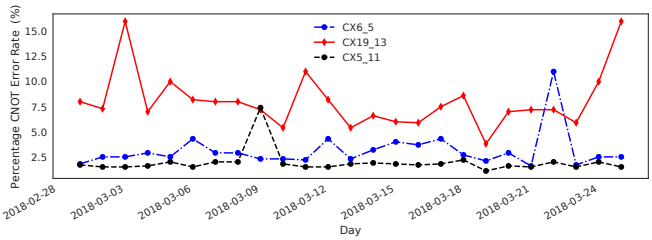


Figure 8. Temporal variation in error rate of two-qubit operations for three links. For most periods, the strong link tends to remain strong and the weak tends to remain weak.

3.5 Spatial Variation in Two-Qubit Gate Errors

Figure 9 shows the layout of the IBM-Q20 qubit computer. Circular nodes represent the qubits and the edges represent a coupling link that is used for performing a two-qubit operation between a pair of qubits. The weight on the edge shows the failure rate of the link and indicates the average probability of failure of the link. For example, the link between $Q14$ and $Q18$ has the highest probability of failure (0.15) and there are several links with a probability of failure as low as 0.02. Thus, there is a variation of 7.5x between the failure rate of the strongest links versus the weakest link.

We observe that for all the metrics we have analyzed (coherence times, error-rate of single-qubit operations, and error-rate of two-qubit operations), there is significant variation in the behavior of qubits and links. Given that the data for this variation can be obtained using characterization (which is performed periodically), we can use the variation data and develop variation-aware policies. We first define our evaluation methodology and the figure of merit (for assessing system level reliability) and then present our proposals.

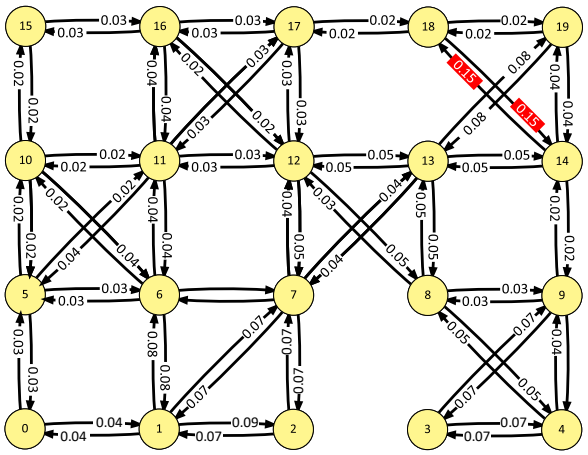


Figure 9. Layout of IBM-Q20, each edge represents a possible 2-qubit operation. The label on the edge represent the average probability of failure on that link when an operation is performed. The best link(s) have an error-rate of 0.02 and the worst link has 0.15, so a difference in strength of 7.5x.

4 Evaluation Methodology

In this section, we define figure-of-merit for system-level reliability, benchmarks, and evaluation infrastructure to estimate the effectiveness of proposed variation-aware policies.

4.1 Figure-of-Merit for System-Level Reliability

In an iterative model of computing for NISQ programs, the trial contributes to useful information if the trail can be executed without errors. In fact, if the workload can be executed with only a small probability of error, then we may not need a large number of trials to converge on the correct solution. To quantify the overall system reliability, we use the *Probability of Successful Trial (PST)* metric as the primary figure-of-merit. *PST* can be computed as the ratio of successful trials to the total number of trials performed.

Table 1. Benchmark Characteristics

NISQ Workload	Benchmark Description	Num Qubits	Total Inst	SWAP Inst
alu	Quantum adder [34]	10	299	19
bv-16	Bernstein Vazirani [3]	16	66	7
bv-20	Bernstein Vazirani [3]	20	90	10
qft-12	Quantum Fourier Trans.	12	344	35
qft-14	Quantum Fourier Trans.	14	550	53
rnd-SD	Rand benchmark with short distance communication	20	100	24
rnd-LD	Rand benchmark long distance communication	20	100	35

4.2 Benchmarks

For our evaluations, we use micro-benchmarks used by the prior studies on quantum compilers and qubit allocation [28, 34] and small kernels demonstrated with IBM quantum computers [5]. These micro-benchmarks are scaled down version of larger quantum applications and subroutines. Table 1 show the seven benchmarks used in our study, their description, number of quantum instruction performed, and the number of qubits and the SWAP operations. We choose workloads with diverse qubit entanglement patterns. For example, Quantum Fourier Transform (qft) requires almost all to all entanglement whereas Bernstein-Vazirani (bv) requires one qubit entangled with rest of the other qubits. Whereas, benchmarks: rnd-SD, rnd-LD have repeated randomized CNOTs.

4.3 Evaluation Infrastructure

We perform our studies using the variation data from the IBM-Q20 machine. Unfortunately, the access to IBM-Q20 is not publicly available. Therefore, for our system reliability evaluations, we built a Monte-Carlo based fault-injection simulator using the architecture-level model for the IBM-Q20 machine. We use the iterative model for NISQ where the same workload is executed a large number of times, and the output is analyzed. Figure 10 shows an overview of our fault-injection simulator.

The simulator accepts the (a) NISQ program (b) layout, configuration, and error rate, and (c) management policies. The simulator injects errors based on the error rate of the given qubit and link and then tracks if the program completed without an error. We use the IBM-Q20 characterization data to estimate the probability of failure for two-qubit, single-qubit, and measurement operations. We perform 1 million trials for each workload to get PST estimates for the NISQ application by modeling errors as uncorrelated events with independent probability across trails.

Note that the compilation and mapping policies can be evaluated by using first-order simulators to gain insights. Nonetheless, in addition to simulation-based evaluations, we also analyze the effectiveness of our proposal on a real quantum machine, albeit at a smaller scale, using the IBM-Q5 machine. In Section 7, we demonstrate that our proposal is effective even in a realistic setting and provides a significant improvement in PST for real systems.

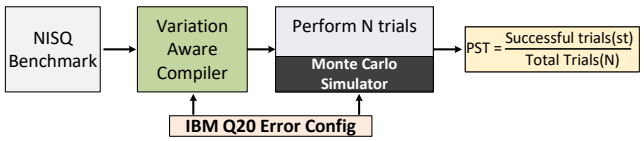


Figure 10. Monte-Carlo fault-injection simulator for estimating system level reliability of quantum computers.

4.4 Layout and Error-Rate Parameters

The layout configuration specifies the number of qubits and their connectivity. For our studies, we use the IBM-Q20 layout and error-rate collected from IBM-Q20 over 52 days as is. The error-rate parameters describe the error rates for single-qubit, two-qubit and measurement operations. We model the errors in quantum operations as independent trials. We also model the coherence errors for all qubits. For the error-rate of IBM-Q20, the gate errors have a domination impact on the overall system reliability and the impact of coherence errors is negligible (e.g., for bv-20, the gate errors are 16x more likely to cause system failures than the coherence errors).

4.5 Baseline for Qubit Movement and Allocation

We use a baseline mapping policy proposed by Zulehner et al. [34] that seeks to minimize the number of SWAP operations. The steps for the baseline scheme are as follows:

1. Initialize an unweighted graph (G) that represents qubits as set of nodes (V) and links as edges (E).
2. Compute distance matrix for minimum number of SWAPs required to entangle any two qubits in G.
3. Partition the input program in layers such that each layer consists of independent operations that can be executed in parallel while respecting data dependencies. For example, an input program is partitioned into L, which is a set of n layers, $L = \{l_0, .., l_i, l_{i+1}, .., l_{n-1}\}$ where n equals the depth of the program.
4. Iterate through all the layers to find the map m_i between program qubit and physical qubit for each layer l_i such that all the CNOTs in the layer can be performed with available physical connectivity.
5. Find optimal set of swap operations ($S_{i \rightarrow i+1}$) for each pair of layers l_i and l_{i+1} that transforms the map m_i to m_{i+1}. To search for the optimal set of SWAPs in an exponentially scaling search space, authors propose to use A* search that using cost function and heuristics based on the number of hops or Manhattan distance.

Note that the baseline tries to reduce the cost of SWAPs by implicitly assuming a uniform cost for all SWAP operations.

5 Variation-Aware Qubit Movement

5.1 The Problem of Qubit-Movement

The Qubit-Movement policy is responsible for deciding the route to take while moving qubit data from one device to another.[1] Such a policy can consider all possible routes and pick the one that requires the fewest number of SWAP instructions. Fortunately, most of the designs for quantum computers use a mesh-like network, so all the choices that go either in the X direction or Y direction towards the destination will have identical Manhattan distance, and hence the identical number of SWAP instructions. For example, for the 6-qubit quantum computer shown in Figure 11, if we want to go from physical qubit A to physical qubit F, all three routes (A-B-C-F, A-D-E-F, A-D-C-F) have identical hop counts (3), and the Qubit-Movement policy can choose any of these routes. It may consider making the Qubit-Movement decision simple by first going in the "X" dimension and then going in the "Y" dimension (or vice versa) – while such a policy would ensure the shortest route (minimum number of SWAP instructions), such a policy would exclude the selection of path A-D-C-F.

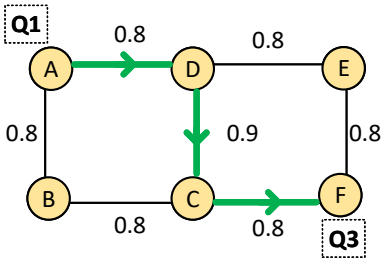


Figure 11. A 6-qubit quantum computer, weight indicates probability of success of each link. To move Q1 (at A) to Q3 (at F), a variation-aware policy would use route A-D-C-F as it maximizes the probability of success of the movement

5.2 "Variation-Awareness" in Qubit-Movement

Given that there is variation in the error-rates of different links, policies (such as X-first or Y-first) that choose one choice among the list of shortest routes will not always provide the best overall system reliability. For example, the number on each link in Figure 11 shows the probability of success of the link. Route A-D-F would maximize the probability of success of the overall movement from A to F, and a variation-aware policy would choose such a route.

[1] Qubit-Movement policy is analogous to network-routing algorithms, which decide the path followed by a packet from the source to destination within a network. Similar to how network-routing algorithms try to minimize the "hop count", Qubit-Movement policies try to minimize the number of SWAPs. Network-routing algorithms make localized decisions at each node, so they must be designed carefully to avoid deadlocks. However, Qubit-Movement is orchestrated globally by the compiler, with the knowledge of the usage of all links, so it is easy to avoid schedules that cause deadlocks.

5.3 Design for Variation-Aware Qubit-Movement

We propose *Variation-Aware Qubit Movement (VQM)* that seeks to perform Qubit-Movement while taking into account the variation in the per-link error rates. VQM selects the paths with the highest reliability for the data movement and actively tries to avoid paths that have poor reliability. Existing mapping policies such as the baseline policy [34] find the optimal path to entangle qubits by formulating a state-space search problem that uses the cost function that is based on the number of inserted SWAPs. Whereas, in VQM, we change the cost function from the number of SWAPs to the overall failure rate incurred by moving the qubit from source to destination. Our variation-aware mapper determines the set of SWAP instructions that minimizes the probability of failure. In case of no variation in error-rates, our policy selects the path with the minimum number of swaps to minimize the probability of failure (identical as a baseline). However, for non-uniform link-errors, VQM picks a path that has the highest reliability. Thus, VQM leverages the locality preserving traits of baseline while using a variation-aware heuristic. Algorithm 1 describes the steps for VQM.

Algorithm 1 Variation-aware qubit movement algorithm

1. Initialize weighted graph (H) with N qubits as vertices (V), links as edges (E) with weights (W) that represent a failure rate of the links, compute distance matrix (D) using Dijkstra's algorithm that holds pairwise shortest distance. Each element in D is the minimum cost for the most reliable path to entangle two qubits on H
2. Using W, compute the node strength or weighted degree (d_i) of each qubit (v_i) such that $d_i = \sum_j^N w_{ij}$
3. Break the input circuit into layers ($l_i = (l_0, l_1, ...l_n)$) similar to baseline.
4. Find program-qubit to physical-qubit map m_i for each layer l_i such that physical qubits with higher node strengths are prioritized during the mapping process.
5. Find optimal set of swap operations that transforms the map m_i to m_{i+1}. The optimal set of SWAPs minimize the probability of error by choosing most reliable paths using D. To choose the optimal set of swaps, we use A* search proposed by the baseline with a reliability-aware cost function and with an additional heuristic Maximum Additional Hop (MAH)

For implementing VQM, we assume that the characterization data of the error rates for different links are available and that this characterization data remains valid during the execution. VQM compiles the application and tries to select the route that tries to maximize system reliability.[2] For selecting the route, VQM simply forms a cost graph where each link has a probability of success, and the overall probability of success of a route is computed as the product of the probability of success of the individual links. VQM selects the route that maximizes the probability of success for the overall route. VQM can select a longer path over the shortest path if the longer path has higher reliability. This will result in an extra number of SWAPs. Furthermore, more qubits get displaced due to a longer chain of SWAPs. The displaced qubits may cause additional set SWAPs for future CNOT operations. We use a parameter that limits the extra SWAP instructions using *Maximum Additional Hop (MAH)*. VQM with such limitations will select the path with the lowest cost, such that the extra hops do not exceed MAH. We use $MAH = 4$ to analyze such hop-limited VQM.

5.4 Impact of VQM on System Reliability

Figure 12 shows the Relative-PST for our seven benchmarks when compiled with VQM and the constrained version of VQM (MAH=4). All benchmarks see a significant improvement in the PST with VQM. Benchmarks such as `qft`, `rnd-LD` require long-distance entanglement and a considerable number of SWAPs (limited locality), therefore they see higher improvement in PST compared to other benchmarks.

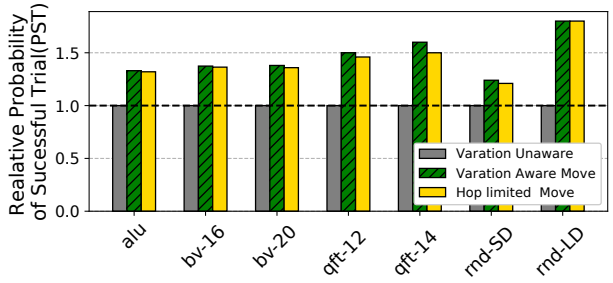


Figure 12. Impact of VQM on the Probability of Successful Trials (PST). Note that reported PST numbers are normalized to PST of the baseline policy that selects the shortest route.

We also observe that the hop-limited policy has similar improvement to an unconstrained policy that does not put any limit on the increased hop count. This is especially true for workloads that have locality as the limited hops preserve the locality by restricting the qubit path among active qubits.

[2]In conventional computer systems, applications may be compiled once, and run unchanged for several years. However, it is reasonable in NISQ domain to assume that each time the workload is scheduled, it gets recompiled by the runtime system (using the latest characterization data) and then repeated trials are performed with the updated executable.

6 Variation Aware Allocation

The Qubit-Allocation policy is responsible for assigning the program qubits to the physical qubits.

6.1 "Variation-Awareness" in Qubit-Allocation

Baseline qubit allocation is oblivious to the variation in the link reliability. It uses an allocation that minimizes the number of SWAPs. For example, if we want to allocate 2 qubits on the machine in Figure 15(a), the baseline policy may pick any two neighboring qubits, including D and A, which are connected by the weakest link. If the allocation policy was aware of the variation, it would pick D and C, which are connected by the strongest link. We propose such a *Variation-Aware Qubit Allocation (VQA)* policy.

6.2 Design of Variation-Aware Qubit-Allocation

The baseline policy starts with carefully selected initial mapping and then tries to converge to a configuration that has a minimum number of SWAPs. However, doing so does not take into account the variation in the link-errors of the qubits. VQA, on the other hand, maps the frequently used qubits to the qubits with most reliable link to improve reliability and preserve the locality. VQA achieves this by starting with the most reliable initial mapping and restricting frequently used pair of qubits to most reliable links. VQA estimates the most frequently entangled qubits by analyzing the first-N instruction in the program and tracking the number of CNOT operations between each of possible pair of qubits. The steps for VQA are shown in Algorithm 2.

Algorithm 2 Variation-aware qubit allocation algorithm

1. Find the sub-graph (SG_k) with k-nodes that has highest aggregate node strength (ANS). $ANS = \sum_i^k d_i$ where $d_i = \sum_j^N w_{ij}$.
2. Find qubit activity by calculating the number of CNOTs per qubit for first t layers.
3. Map program qubit to physical qubit mapping m_i prioritize the mapping of qubits with high activity to SG_k such that top K active qubits are mapped to SG_k.
4. Use baseline algorithm to find SWAPs between layer l_i and l_{i+1}.

Furthermore, when mapping less number of qubits than the available qubits, baseline exposes all the qubits to the mapping process which can sometimes result in the mapping of frequently used qubits to weak qubits. VQA prevents such undesirable assignments by selecting the strongest (subgraph) and restricting the qubit mapping that maximizes the overall system reliability. VQA computes the strongest set of sub-graphs by using K-core algorithm that that recursively prunes nodes with degrees less than k [2].

6.3 Impact of VQA on System Reliability

By using VQA, we ensure the mapping of program qubits with high activity (total number of CNOT operations) to the set of physical qubits with the higher node strength. This improves the reliability for workloads that has repeated entanglement operations between few select pairs of qubits. We implement VQA in conjunction with the variation-aware movement. Figure 13 shows the relative-PST for the micro-benchmarks normalized to the baseline, VQM, and VQM+VQA. Our evaluations show that VQM+VQA can provide up to 1.7x improvement in PST. Note that, for all the benchmarks, the combination of VQM+VQA provides higher PST than the VQM scheme standalone.

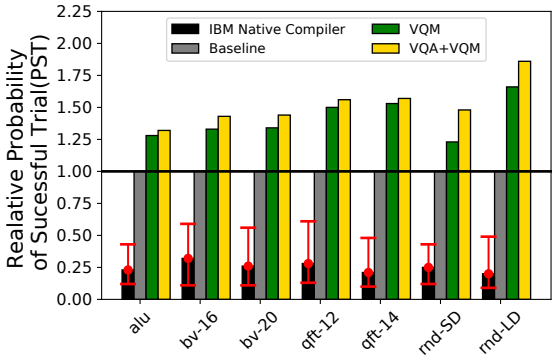


Figure 13. PST for VQA and VQM+VQA normalized the baseline policy (variation unaware). We also compare the normalized PST of the native compiler of IBM.

6.4 Improvement Relative to Native IBM Compiler

We use a state-of-the-art baseline policy that tries to minimize the number of SWAP instructions. Our baseline is stronger than an alternative policy that uses randomized assignment, such as the native compiler from IBM. Figure 13 compares the PST for IBM's native compiler with our baseline and the proposed policies. As the IBM native compiler performs randomized initial mapping, we evaluate 32 configurations (each over 10000 trials) and report the average and the min-max (using the error-bars) PST. Note that our baseline policy has 4x higher PST than the IBM native compiler. Whereas, VQA+VQM improve PST up to 7x over the IBM compiler.

6.5 Effectiveness to Per-Day Variation

We perform our evaluations using average behavior of the link/qubit based on characterization data across 52 days. The behavior of the qubit and links can vary over time, and with it the benefit of our scheme. To analyze this, we evaluated `bv-16` with per-period characterization data across the 52 days. Figure 14 shows the improvement in PST for `bv-16` benchmark for each day (the dotted line denotes the average). VQA+VQM provides larger PST improvement on days with higher variability and smaller on days with lower variability.

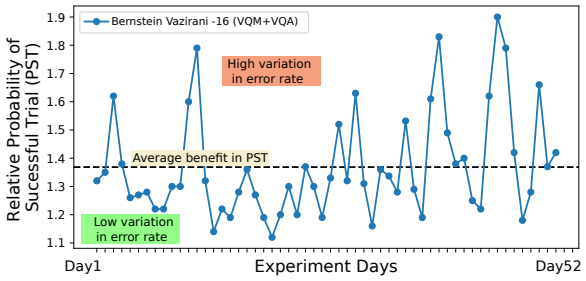


Figure 14. Relative improvement in PST for bv-16 benchmark evaluated with error-configs collected over 52 days

6.6 Sensitivity to Scaling of Error Rates

As technology improves, we can expect the error rates to reduce, however the variation may still persist even at lower error rates, meaning our proposal can still be effective. We evaluate `bv-16` benchmark with 10x lower average error rate (standard deviation reducing proportionally or by half as much). As shown in Table 2, VQM+VQA provides significant benefits that increases with increased relative variation.

Table 2. Sensitivity of VQA+VQM with Error Scaling.

Benchmark Name	Average Error Error-Rate	Covariation of Error Rate	Relative PST Benefit(VQA+VQM)
`bv-16`	1x	Cov-Base	1.43x
`bv-16`	10x lower	Cov-Base	2.02x
`bv-16`	10x lower	2*Cov-Base	2.59x

7 Evaluation on Real System: IBM-Q5

Access to IBM-Q20 is not publicly available, so we evaluated our policies for IBM-Q20 using a simulator. We demonstrate the usefulness of our ideas for real quantum systems by performing experiments on the IBM-Q5 machine. For IBM-Q5, the average two-qubit error rate is 4.2%, and the worst link-error is 12%. We use the error configuration of IBM-Q5 to compile the benchmarks that are suitable for IBM-Q5. A compiled program with the baseline policy and with VQA+VQM are then executed on a IBM-Q5 machine and the output is logged. We run each experiment with 4096 trials and analyze the output log to compute the PST for each program and policy. Table 3 shows the PST of the baseline and (VQA+VQM). Our proposal improves the PST for IBM-Q5 machine by up to 1.9x improvement (average 1.36x).

Table 3. PST for Baseline and Proposed Policies on IBM-Q5.

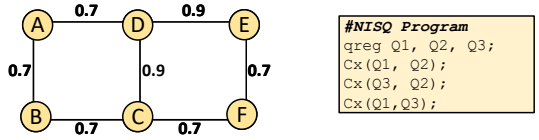
Benchmark Name	PST (Baseline)	PST (VQA+VQM)	Relative Benefit in PST
`bv-3`	0.31	0.38	1.22x
`bv-4`	0.21	0.23	1.09x
`TriSwap`	0.13	0.25	1.90x
`GHZ-3`	0.57	0.77	1.35x
`GeoMean`	0.26	0.36	1.36x

8 Partitioning Quantum Computer

We have explored the variation-aware policies for Qubit-Movement and Qubit-Allocation. This concept can be used to provide insights into other design trade-offs that may come in NISQ systems. We do a case study for a scenario, where the workload requires half or fewer qubits than what is physically available, and the computer can be partitioned to run multiple copies of the same workload (to provide more trials per unit time). We analyze whether it makes sense to partition the NISQ computer in such scenarios.

8.1 Two Weak-Copies versus One Strong-Copy

When the number of program qubits are less than or equal to half of the physical qubits, we can run two copies of the same program. In an ideal world, the simultaneously running two copies can provide twice as many number of error-free trials per unit time. However, for a quantum computer with variation, running two copies restrict the program qubit to physical qubit mappings. For example, running a single copy provide an opportunity to choose the strongest set of qubits and links in a given quantum computer, whereas, running two copies would constrain us to also use weaker set of qubits and links. Thus, the single copy would try to maximize the PST for a given trial, even if it means sacrificing the increased trials per unit time that would be possible with two copies. Whereas, having two-copies provides more trials per unit time at the expense of PST for each trial. On a given NISQ with variable reliability, should we run two weak copies or run one strong copy of the program?



(a)

```
#NISQ Program
qreg Q1, Q2, Q3;
Cx(Q1, Q2);
Cx(Q3, Q2);
Cx(Q1,Q3);
```

Map Q_{1X}, Q_{2X}, Q_{3X} → A, B, C Map Q_{1Y}, Q_{2Y}, Q_{3Y} → D, E, F	
Copy-X	Copy-Y
Cx (A,B) -- 0.7	Cx (D,E) --0.9
Cx (B,C) -- 0.7	Cx (E,F) --0.7
SWAP (B,C) -- (0.7)^3	SWAP (D,E) -- (0.9)^3
Cx (A,B) -- 0.7	Cx (E,F) -- 0.7
PST(X) = 0.12	PST(Y) = 0.32

(b)

Map Q_{1X}, Q_{2X}, Q_{3X} → C, ,D, E Map Q_{1Y}, Q_{2Y}, Q_{3Y} → DON'T MAP	
Single Copy	NA
Cx (A,B) -- (0.9)	--
Cx (B,C) -- (0.9)	--
SWAP (B,C) -- (0.9)^3	--
Cx (A,B) -- (0.9)	--
PST = 0.53	NA

(c)

Figure 15. (a) NISQ with six qubits using mesh connectivity. A CNOT reliability is reported on the top of each link. (b) (c) Mapping policy that runs two copies of a NISQ program (d) Mapping policy that runs one copy using the strongest links

For a hypothetical machine with six physical qubits with a mesh-layout as shown in the Figure 15(a). The edge-weights in the graph show the strength of the coupling links. For a quantum program with three program qubits as shown in the Figure 15(a), we can either run two copies by partitioning the quantum computer or run just one copy. Figure 15(b), shows two copies of a program: Copy-X and Copy-Y running on a quantum computer. The success probability of individual copy can be calculated by multiplying all the success probabilities of operations in the program. For example, Figure 15(b) shows the PST for Copy-X and Copy-Y to be 0.32 and 0.12 respectively. Thus, running two copies does not increase the rate at which successful trials can be done by 2x, instead in our case it is only 37.5% (0.44/0.32).

For the example program, if we choose to run a single copy, we can intelligently select the strongest subset of qubits and links to improve the overall reliability. Figure 15(c) shows one such example whereby choosing to run just one strong copy can improve the cumulative PST. When running two copies, the constraints on connectivity restricts the use of link CD which is one of the strongest links. When running two copies, programmer has to resort to the weaker links. Whereas, when running a single copy, we can pick most reliable links and achieve better PST as shown in the Figure 15(b).

8.2 Benchmark-Based Evaluation

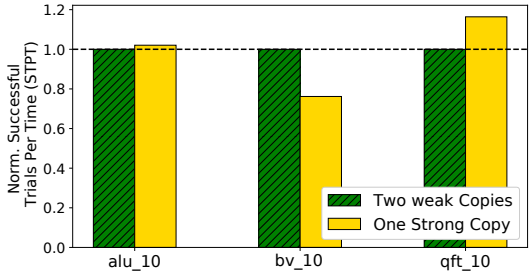


Figure 16. Successful Trials per Unit Time (STPT) when running (a) two-copies (b) one strong copy. Note that the micro-benchmarks were modified to have 10 program qubits.

We extend our evaluation infrastructure to support two copies of the same workload. For the two-copy mode, we explore all possible partitions and select the best. Note that besides the number of copies, movement and the mapping algorithm used for both of the policies are identical. The only difference is the available number of qubits. For the evaluation in this section, we use the figure of merit as *Number of Success Trials Per Unit Time (STPT)*, as it captures both the PST and the increased rate of trials with two copies. We modify these benchmarks to use only 10 qubits. Figure 16 shows the STPT of the single strong-copy and two-copies, both normalized to the STPT of the two copies. For this study, we selected the three workloads that can operate with ten qubits. We observe that sometimes two-copy is better (`bv-10`) and sometimes one strong-copy is better (`qft-10`). For NISQ applications, we can estimate which solution is likely to perform better for the workload and used that solution. Thus, our variation-aware policies may be useful in enabling *Adaptive Partitioning* for NISQ machines, where the decision between one strong copy versus two-copies can be based on STPT.

9 Discussion of Limitations

Computer architecture for NISQ-era quantum computers is still in the stage of infancy. There is no clear and established evaluation infrastructure to estimate the impact of different policies on the system-level reliability of quantum computers using benchmarks or applications, especially when direct-access to the quantum machine is not available. As with any initial research, our study is based on a number of assumptions, which may not hold, as the technology matures. We discuss some of the limitations of our study:

Workloads: Our evaluations are done using small kernels and random benchmarks, similar to the ones used in the area of compilation for quantum computers. These kernels and benchmarks may not be representative of the NISQ applications that will be developed in the future.

Error Models: We make several simplifying assumptions such as no-correlations between errors, static error-rates, and exponential-model for coherence errors. Noise in the real quantum computer is significantly complex to model and is currently an open problem.

The basic insight in our work is that there is variation in qubit and link reliability. Exploiting the variability allows better-than-worst-case behavior and avoids the overall system reliability getting dictated by a few weak components. While we expect this basic insight is useful for future quantum computers, some of our assumptions about evaluations and error models may get redefined as the field progresses.

10 Related Work

Quantum System Architecture: Early works in quantum system architecture provided a blueprint for quantum systems by defining system abstractions [1, 14, 29, 31]. Several papers highlighted the resource overheads of quantum error correction and proposed the microarchitectural solutions to mange error correction efficiently [12, 23]. A large body of work has also focused on quantum compilers that synthesize, simulate, and analyze quantum programs [13, 18].

NISQ Compilers: The availability of NISQ machines to the general public has sparked the interest in building compiler tools for the near-term quantum computers. Moreover, recent works provide an excellent theoretical understanding of the mapping problems [4, 8, 20, 26, 28, 32]. Along with general mapping problems, researchers have started focusing on the machine specific mapping problems [28, 34].

Reliability Metrics: IBM researchers proposed the *Quantum Volume (QV)* metric to compare quantum computers with different qubit technologies and varying degree of connectivity. However, QV does not capture the reliability loss due to variation, is an application-agnostic metric, and does not account for policy decisions. Therefore, we use PST as the critical metric for system reliability [19].

Demonstrations on IBM Quantum computers: IBM Quantum Experience is a cloud service that enables public access to quantum computers [5]. Researchers have used these insights to design light-weight error detection/correction codes or device level techniques that can improve the gate fidelity [7, 9, 24]. Variation in error-rate is a crucial hurdle towards developing new applications. Currently, a programmer has to rely on hand optimized data-movements and mapping to improve the application-level reliability [33]. Unfortunately, the error rate and error patterns are not static, and they change every calibration cycle (performed twice a day for IBM computers), rendering hand-optimized mappings unscalable.

11 Summary

We study the policies for *Qubit-Allocation* and *Qubit-Movement* for current quantum computers. We observe that there can be variation in the error rates of different qubits and links, which can mean that prior studies that try to minimize communication may not maximize overall system reliability. The system reliability of quantum computers can be improved significantly by steering more operations towards stronger qubits and links and limiting operations on weak links. To this end, our paper makes the following contributions:

- We highlight the problem of variability in error rates by analyzing the publicly available characterization data for IBM-Q20. We show that there is significant variation in the error rate of qubits and links.
- We propose *Variation-Aware Qubit Movement* policy that exploits the variation in error rates by trying to pick a route that has the lowest probability of failure.
- We propose *Variation-Aware Qubit Allocation* policy that exploits the variation in error rates by allocating program qubits to physical qubits such that the use of links with high error rates gets minimized.
- We develop an evaluation methodology to assess the impact of device variation and management policies on the system-level reliability of quantum computers. We show that our policies provide significant improvement both in simulated setting and on the IBM-Q5.

Our insights can also help in understanding the resource sharing and partitioning problems in the near-term quantum computers, such as deciding between running one strong-copy versus two concurrent copies of NISQ program.

Acknowledgments

An earlier version of this paper [30] was submitted to MICRO 2018. The primary change in this version is the addition of Section 7, evaluation on a real-quantum system.

We thank Poulami Das for editorial feedback. This work was supported by a gift from Microsoft Research.

References

[1] Steven Balensiefer, Lucas Kregor-Stickles, and Mark Oskin. An evaluation framework and instruction set architecture for ion-trap based quantum micro-architectures. In *Proceedings of the 32nd Annual International Symposium on Computer Architecture*, ISCA '05, pages 186–196, Washington, DC, USA, 2005. IEEE Computer Society.

[2] Vladimir Batagelj and Matjaz Zaversnik. An o (m) algorithm for cores decomposition of networks. *arXiv preprint cs/0310049*, 2003.

[3] Charles H Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. Strengths and weaknesses of quantum computing. *SIAM journal on Computing*, 26(5):1510–1523, 1997.

[4] Kyle EC Booth, Minh Do, J Christopher Beck, Eleanor Rieffel, Davide Venturelli, and Jeremy Frank. Comparing and integrating constraint programming and temporal planning for quantum circuit compilation. *arXiv preprint arXiv:1803.06775*, 2018.

[5] International Business Machines Corporation. Universal Quantum Computer Development at IBM:. http://research.ibm.com/ibm-q/research/, 2017. [Online; accessed 3-April-2017].

[6] Michel H Devoret and Robert J Schoelkopf. Superconducting circuits for quantum information: an outlook. *Science*, 339(6124):1169–1174, 2013.

[7] Davide Ferrari and Michele Amoretti. Demonstration of envariance and parity learning on the ibm 16 qubit processor. *arXiv preprint arXiv:1801.02363*, 2018.

[8] Gian Giacomo Guerreschi and Jongsoo Park. Two-step approach to scheduling quantum circuits. *Quantum Science and Technology*, 2018.

[9] Robin Harper and Steven Flammia. Fault tolerance in the ibm q experience. *arXiv preprint arXiv:1806.02359*, 2018.

[10] Matthew B. Hastings, Dave Wecker, Bela Bauer, and Matthias Troyer. Improving quantum algorithms for quantum chemistry. *Quantum Info. Comput.*, 15(1-2):1–21, January 2015.

[11] Jeremy Hsu. CES:Intel's 49-Qubit Chip Shoots for Quantum Supermacy. https://spectrum.ieee.org/tech-talk/computing/hardware/intels-49qubit-chip-aims-for-quantum-supremacy, 2017. [Online; accessed 3-April-2018].

[12] Nemanja Isailovic, Mark Whitney, Yatish Patel, and John Kubiatowicz. Running a quantum circuit at the speed of data. In *ACM SIGARCH Computer Architecture News*, volume 36, pages 177–188. IEEE Computer Society, 2008.

[13] Ali JavadiAbhari, Shruti Patil, Daniel Kudrow, Jeff Heckey, Alexey Lvov, Frederic T Chong, and Margaret Martonosi. Scaffcc: Scalable compilation and analysis of quantum programs. *Parallel Computing*, 45:2–17, 2015.

[14] N Cody Jones, Rodney Van Meter, Austin G Fowler, Peter L McMahon, Jungsang Kim, Thaddeus D Ladd, and Yoshihisa Yamamoto. Layered architecture for quantum computing. *Physical Review X*, 2(3):031007, 2012.

[15] Julian Kelly. A Preview of Bristlecone, GoogleâĂŹs New Quantum Processor. https://research.googleblog.com/2018/03/a-preview-of-bristlecone-googles-new.html, 2018. [Online; accessed 3-April-2018].

[16] Will Knight. IBM Raises the Bar with a 50-Qubit Quantum Computer. https://www.technologyreview.com/s/609451/ibm-raises-the-bar-with-a-50-qubit-quantum-computer/, 2017. [Online; accessed 3-April-2018].

[17] Emanuel Knill, D Leibfried, R Reichle, J Britton, RB Blakestad, JD Jost, C Langer, R Ozeri, Signe Seidelin, and David J Wineland. Randomized benchmarking of quantum gates. *Physical Review A*, 77(1):012307, 2008.

[18] Daniel Kudrow, Kenneth Bier, Zhaoxia Deng, Diana Franklin, Yu Tomita, Kenneth R Brown, and Frederic T Chong. Quantum rotations: a case study in static and dynamic machine-code generation for quantum computers. In *ACM SIGARCH Computer Architecture News*, volume 41, pages 166–176. ACM, 2013.

[19] Norbert M Linke, Dmitri Maslov, Martin Roetteler, Shantanu Debnath, Caroline Figgatt, Kevin A Landsman, Kenneth Wright, and Christopher Monroe. Experimental comparison of two quantum computing architectures. *Proceedings of the National Academy of Sciences*, 114(13):3305–3310, 2017.

[20] Dmitri Maslov, Sean M Falconer, and Michele Mosca. Quantum circuit placement: optimizing qubit-to-qubit interactions through mapping quantum circuits into a physical experiment. In *Proceedings of the 44th annual Design Automation Conference*, pages 962–965. ACM, 2007.

[21] Tzvetan S Metodi, Arvin I Faruque, and Frederic T Chong. Quantum computing for computer architects. *Synthesis Lectures on Computer Architecture*, 6(1):1–203, 2011.

[22] Andrea Morello and David Reilly. What would you do with 1000 qubits? *Quantum Science and Technology*, 3(3):030201, 2018.

[23] Mark Oskin, Frederic T Chong, and Isaac L Chuang. A practical architecture for reliable quantum computers. *Computer*, 35(1):79–87, 2002.

[24] Bibek Pokharel, Namit Anand, Benjamin Fortman, and Daniel Lidar. Demonstration of fidelity improvement using dynamical decoupling with superconducting qubits. *arXiv preprint arXiv:1807.08768*, 2018.

[25] John Preskill. Quantum computing in the nisq era and beyond. *arXiv preprint arXiv:1801.00862*, 2018.

[26] Alireza Shafaei, Mehdi Saeedi, and Massoud Pedram. Optimization of quantum circuits for interaction distance in linear nearest neighbor architectures. In *Proceedings of the 50th Annual Design Automation Conference*, page 41. ACM, 2013.

[27] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

[28] Marcos Siraichi, Vinicius Fernandes Dos Santos, Sylvain Collange, and Fernando Magno Quintão Pereira. Qubit allocation. In *CGO 2018-IEEE/ACM International Symposium on Code Generation and Optimization*, pages 1–12, 2018.

[29] Krysta M Svore, Alfred V Aho, Andrew W Cross, Isaac Chuang, and Igor L Markov. A layered software architecture for quantum computing design tools. *Computer*, 39(1):74–83, 2006.

[30] Swamit S Tannu and Moinuddin K Qureshi. A case for variability-aware policies for nisq-era quantum computers. *arXiv preprint arXiv:1805.10224*, 2018.

[31] Rodney Van Meter and Clare Horsman. A blueprint for building a quantum computer. *Commun. ACM*, 56(10):84–93, October 2013.

[32] Davide Venturelli, Minh Do, Eleanor Rieffel, and Jeremy Frank. Compiling quantum circuits to realistic hardware architectures using temporal planners. *Quantum Science and Technology*, 3(2):025004, 2018.

[33] Yuanhao Wang, Ying Li, Zhang-qi Yin, and Bei Zeng. 16-qubit ibm universal quantum computer can be fully entangled. *arXiv preprint arXiv:1801.03782*, 2018.

[34] Alwin Zulehner, Alexandru Paler, and Robert Wille. Efficient mapping of quantum circuits to the ibm qx architectures. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2018*, pages 1135–1138. IEEE, 2018.