# Experimental Evaluation of NISQ Quantum Computers: Error Measurement, Characterization, and Implications[*]

Tirthak Patel, Abhay Potharaju, Baolin Li, Rohan Basu Roy, Devesh Tiwari
Northeastern University

*Abstract*—**Noisy Intermediate-Scale Quantum (NISQ) computers are being increasingly used for executing early-stage quantum programs to establish the practical realizability of existing quantum algorithms. These quantum programs have uses cases in the realm of high-performance computing ranging from molecular chemistry and physics simulations to addressing NP-complete optimization problems. However, NISQ devices are prone to multiple types of errors, which affect the fidelity and reproducibility of the program execution. As the technology is still primitive, our understanding of these quantum machines and their error characteristics is limited. To bridge that understanding gap, this is the first work to provide a systematic and rich experimental evaluation of IBM Quantum Experience (QX) quantum computers of different scales and topologies. Our experimental evaluation uncovers multiple important and interesting aspects of benchmarking and evaluating quantum program on NISQ machines. We have open-sourced our experimental framework and dataset to help accelerate the evaluation of quantum computing systems.**

*Index Terms*—**Quantum Computing, Performance Evaluation, Computer Errors, Error Analysis, Error Probability**

## I. Introduction

Quantum computing is maturing at a fast pace, moving from theoretical ideas to practical implementations. Microsoft recently announced their topological quantum bit technology, Google a quantum advantage using their 53-qubit Sycamore quantum machine and IBM became the first company to take quantum computing to the cloud by providing the first publicly-available superconducting-qubits quantum machines.

Unfortunately, current Noisy Intermediate-Scale Quantum (NISQ) machines suffer from high error rates and do not have enough number of qubits to deploy Error Correction Codes (ECC) [36, 48]. But, NISQ machines employ a circuit-based approach toward developing and executing quantum programs, including practical high-performance computing (HPC) problems [6, 24, 36, 48], as compared to the quantum annealing approach which is limited to a subset of problems [22, 42, 43]. However, because of the relatively high error rate of NISQ machines, the output generated by these programs is erroneous and challenging to reproduce.

[*]A part of this work was performed during an unprecedented time – around the peak of the COVID-19 pandemic. We were able to conduct this scientific study only because first-line responders and essential workers worked tirelessly to keep our community safe and functioning – we are eternally grateful to them. This paper is dedicated to the memories of all the first-line responders and essential workers who sacrificed their lives trying to keep ours safe.

To overcome this challenge and to effectively use NISQ machines, it is critical to have a deep understanding of the properties of different operations and real-world programs on current state-of-art quantum machines. Previous works in the area of NISQ computing have mostly focused on four major research aspects: (1) quantum simulation and experimental frameworks [9, 20, 31, 60], (2) optimal qubit mapping of a quantum program to a quantum machine given its error probability [3, 18, 32, 33, 40, 50, 51, 53–55, 59, 63], and (3) execution of multiple codes in parallel [14, 41, 62], and (4) debugging quantum programs and estimating correct outputs [34, 39, 46]. *However, no previous work has taken a data-driven approach to provide a detailed characterization of error and execution time characteristics of quantum programs on real quantum machines.*

**To the best of our knowledge, this is the first systematic and rich experimental evaluation effort that measures and analyzes the error and execution time characteristics of seven IBM Quantum Experience (QX) quantum machines of different scales and topologies**. Our evaluated quantum benchmarks solve problems from different scientific and optimization domains, and our experimental study covers multiple months of characterization data for seven IBM QX cloud quantum machines. Our study is open-sourced at `https://doi.org/10.5281/zenodo.3957894`.

Below, we summarize a subset of our findings, their implications and how our open-source code and dataset can be used by the high-performance quantum computing to advance the quantum computing field further.

• Our benchmarking effort, which spans many months and covers a wider variety of quantum computing platforms, confirms the swift progress in quantum computing technology. As expected, the error rates for different quantum operations and coherence have improved, but readout errors continue to be a bottleneck, this is primarily because readout operations are susceptible to the signal amplification noise.

• Our experimental results reveal that although quantum computing technology has been improving significantly, the optimal quantum machine for different types of quantum operations is not necessarily the latest one. Moreover, different quantum programs have their lowest output error on different quantum machines based on the composition of their gate and

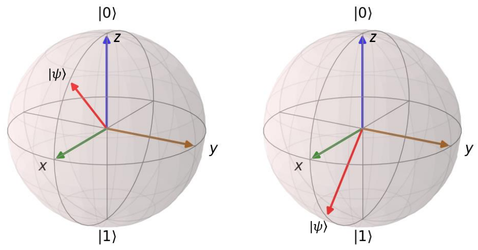


**Fig. 1.** Bloch sphere of the state $|\Psi\rangle$ of a qubit before applying a Pauli-X gate (left) and after applying a Pauli-X gate (right).

readout operations. Also, this choice may potentially change over time across machines (requiring additional careful description of experimental methodology compared to the classical computing domain). Our open-source benchmarking data will enable end-users to choose a better platform for their programs based on our insights about what kind of operations are more suited for which platform.

• Currently, the state-of-the-art research proposals for error-aware circuit mappings [3, 32, 51, 53] only take the average or most recently reported error rate for different qubits into considerations. In practice, we find that error variance and frequently changing error characteristics also play a major role in determining the degree of error in the final program output. We define “operation quality” as a new metric to capture this behavior and find that some qubits/operations are more unstable than others..

• Our results show that the quantum machine that has the lowest execution times, does not necessarily have the lowest output errors for a given quantum program. Additionally, selecting higher compiler optimization levels does not necessarily improve the output error or the execution time of the programs. Surprisingly, reducing the circuit depth and the number of gates of a circuit increases the variability of the output error. Thus, quantum programmers need to choose the compiler optimizations carefully and report the optimizations in the results.

• We discover that the output states of a quantum program with lower hamming weights tend to experience higher output errors due to the accumulation of errors from states with higher hamming weight. A useful implication for quantum programmers is to map the important states of the program output carefully to the states with higher hamming weights.

## II. Quantum Computing Background

This section briefly describes the basics of quantum computing and the technology behind quantum machines at IBM.

**Brief Background on Quantum Computing.** The fundamental unit of quantum computing is a quantum bit or qubit. A qubit has two basis states denoted as $|0\rangle$ and $|1\rangle$. In contrast to a classical bit, which only takes the value of either 0 or 1, a

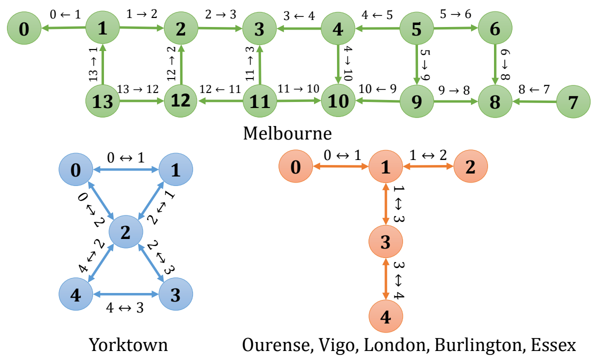


**Fig. 2.** Architecture of IBM QX quantum computers. Each circle represents a qubit with the corresponding Id. The arrow directions indicate the relationship from control qubits to target qubits for 2-qubit gate operations.

**TABLE I.** IBM QX machines.

| **Online date** | **Machines (Number of Qubits)** |
|---|---|
| Nov 06, 2018 | Melbourne (14), Yorktown (5) |
| Jul 03, 2019 | Ourense (5), Vigo (5) |
| Sep 13, 2019 | London (5), Burlington (5), Essex (5) |

qubit can also be in a *superposition* of the two states. A qubit state is represented as $|\Psi\rangle = a\,|0\rangle + b\,|1\rangle$, where $a$ and $b$ are complex numbers and $\|a\|^2 + \|b\|^2 = 1$. When a qubit is *read out*, it collapses into one of the two basis states: state $|0\rangle$ with probability $\|a\|^2$ and $|1\rangle$ with probability $\|b\|^2$.

A qubit's state can be considered as a point on a unit sphere called a Bloch sphere. In Fig. 1, each sphere represents a qubit state. While classical computing bits can only be switched between the north pole and the south pole, quantum gates can be used to apply rotation transformations to change the state of a qubit to anywhere on the sphere. The Pauli-X gate and Hadamard gate are examples of 1-qubit gates. As shown in Fig. 1, a Pauli-X gate rotates the qubit around the x-axis by $\pi$ radians, similar to a classical NOT gate. A Hadamard gate puts each of the basis states into an equal superposition of $|0\rangle$ and $|1\rangle$. Multi-qubit gates facilitate *entanglement*: two or more qubits are coupled such that the state of each qubit cannot be described independently of the state of others. The Controlled-X (*CX*) gate is an example of a 2-qubit gate. It has a control qubit and a target qubit. A Pauli-X gate is applied to the target qubit when the control qubit is in the $|1\rangle$ state.

Unfortunately, a qubit can retain its state only for a limited amount of time. The period for a qubit's natural decay from excited state $|1\rangle$ to ground state $|0\rangle$ is called **T1 coherence time** (amplitude damping). The period for a qubit's state change due to environment interaction is called **T2 coherence time** (phase damping). Longer coherence time is essential to the device's performance because a more significant number of operations can be accomplished before the output becomes erroneous beyond a tolerance limit [15]. Qubits are error-prone because of high volatility and susceptibility to environmental perturbations. The **readout error** is the probability of incorrect

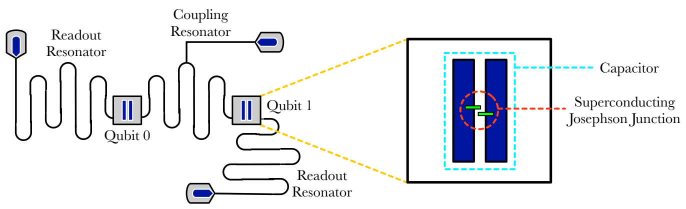


**Fig. 3.** Design of IBM's superconducting qubit technology (from Patel et al. [44]).

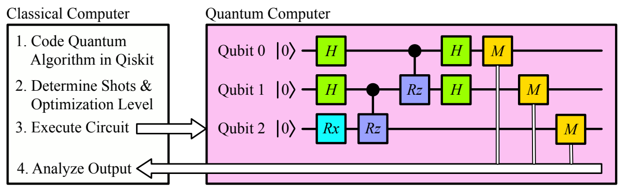


**Fig. 4.** Execution of an example quantum program (Quantum Phase Estimation (QPE)) mapped to a quantum circuit.

measurement of a qubit state (referred to as 1-qubit readout operation). The **gate error** is the probability of introducing an error during a gate operation, for example, rotating a state of a qubit by a slightly erroneous angle.

**Quantum Computing at IBM QX.** Multiple quantum machines have been made available to the public through the IBM QX Cloud. Table I lists all the IBM QX machines used in this study and their online launch dates. Melbourne has 14 qubits, while other machines have 5 qubits. The QX architecture supports the 2-qubit CX gate and a set of three 1-qubit gates: U1, U2, and U3. The U3 gate is a universal 1-qubit gate and can be represented as $U3(\theta, \phi, \lambda) = R_z(\phi)R_y(\theta)R_z(\lambda)$. Any 1-qubit operation can be conducted by specifying the input parameters $\phi$, $\theta$, $\lambda$. The U1 and U2 gates are special cases of the U3 gate. The QX architecture is considered a universal quantum computing architecture because all gates, including gates spanning more than two qubits (such as a *Toffoli gate*), are broken down into U1, U2, U3 and CX gates to perform hardware operations.

The architectures of the studied machines are shown in Fig. 2. Melbourne and Yorktown, which were made available in 2018, have a different layout than the other machines. To map any complex quantum gate operation to the machine, we need to decompose it into the elementary *CX* and *U3* gates. Its unique architecture restricts the mapping of a 2-qubit gate to a machine. For example, on Vigo, we cannot directly map the gate to qubits 2 and 3 because they do not share a direct connection. On Melbourne, we cannot directly map the control qubit to 0 and target qubit to 1 because it violates the control direction for this pair of qubits.

On IBM's quantum machines, the qubits are implemented using Josephson Junctions created by separated superconducting electrodes and capacitors as shown in Fig. 3. One-qubit gate operations are performed by applying external controls in the form of microwave pulses. Errors in applying these pulses cause 1-qubit gate errors. Entanglement between two qubits is performed using coupling resonators. These coupling resonators can be error-prone, causing 2-qubit gate errors. Lastly, qubit state measurement (or readout operation) is performed using readout resonators as shown in Fig. 3. The readout resonators are also error-prone and can cause 1-qubit readout errors when qubit states are measured.

## III. Experimentation Methodology

This section describes our experimental methodology.

**Obtaining Properties of Qubits and Quantum Operations.** We obtain the *properties* of all qubits and operations across all available IBM quantum machines by querying the IBM QX cloud using IBM's Qiskit framework [2]. These properties include T1 and T2 coherence times, and gate and readout error probabilities and execution times. The properties frequently change as machines are calibrated at least once daily to minimize operation error probabilities. So, we collected data daily for over two months.

**Methodology of Executing a Quantum Program in IBM QX.** A *quantum program* is expressed as a sequence of gate operations operating on multiple qubits. The mapping of a quantum program on a quantum machine is called a *quantum circuit*: a set of quantum operations and the corresponding qubits on which the operation is performed. As shown in Fig. 4, running a quantum program requires several tasks to be performed on both classical and quantum machines.

First, the program is coded using IBM's Qiskit language. Then, to run the program on a quantum machine, we tune several parameters to optimize the output error and execution times: (1) The number of "shots" is the number of times the program is run on the quantum machine. Because the output of a quantum program is probabilistic, it needs to be run several times (e.g., 1024 times) to get the probabilities of each output state. We perform experiments with 4 shot levels (1024, 2048, 4096, and 8192) shots for all programs. (2) The optimization level determines the types of compiler optimizations that are applied when a program is mapped to a quantum computing hardware. Note that a *single* quantum program can map to *multiple* "circuits" in different ways on the same machine. Optimizations such as the selection of less error-prone qubits, selection of operations to reduce the "depth" (maximum number of serially performed operations) of a circuit, and reducing the number of total operations can be performed based on the selected level. We run all programs for each of the 4 possible compiler optimization levels (0, 1, 2, and 3). At optimization level 0, the circuit is mapped to the machine without any optimizations. At level 1, light optimization of collapsing adjacent gates into fewer gates is performed. At level 2, optimizations are performed based on gate commutation relationships and error-aware mapping. Lastly, at level 3, along with all the previous optimizations,

**TABLE II.** Quantum programs used for analysis.

| Program ID | Program description |
|---|---|
| BV | Bernstein-Vazirani algorithm [5] |
| DJ | Deutch-Josza algorithm [11] |
| QFT | Quantum Fourier transform [16] |
| QPE | Quantum phase estimator [10] |
| QAOA | Quantum Alternating Operator Ansatz [17] |
| GROVER | Grover's algorithm [19] |
| 3SAT | 3-clauses satisfiability solver [25] |
| CHEM | Simulation using Variational Quantum Eigensolver [38] |
| SIMON | Simon's algorithm [29] |

re-synthesis of two-qubit blocks of gates in the circuit is performed to further reduce the number of operations.

Next, the circuit is run on a quantum machine, as shown in Fig. 4. Each horizontal line represents a qubit, and each box represents a quantum operation. Left to the right indicates the time order of the operations performed. In the Quantum Phase Estimation (QPE) program circuit shown here, $H$ is the Hadamard gate, $R_x$ and $R_z$ are Bloch sphere rotations about the respective axes, and $M$ is the readout operation at the end. Once the output is read out, it is returned to the classical machine where it can be analyzed. Qiskit returns program information such as circuit map, the number of gates, depth, and output state probabilities.

We ran all programs daily multiple times on all available IBM machines over two months. However, due to the long queues on each machine to perform a job, we avoided running on machines with job queues longer than 25 pending jobs to be able to collect the results frequently.

**Selection of Quantum Programs.** Table II summarizes the programs used for our analysis. We choose our analysis programs based on their application in the field of quantum computing as well as future scope for adaptations of these programs to solve classical high-performance computing problems. For instance, Simon, Deutsch-Josza, and Quantum Phase Estimation programs have an important role in inspiring the Quantum Fourier Transform program. Many HPC applications, such as those in the Exascale Computing Project (ECP) and HPC Challenge (HPCC )benchmark suites [35, 49], rely on fast Fourier Transform implementations. A quantum implementation can significantly speed up these programs. Moreover, Bernstein-Vazirani and Deutsch-Josza programs are important programs used to show the merits of using a quantum machine for specific problems as they can bring exponential improvement over existing classical methods. Quantum programs such as Variational Quantum Eigensolver-based molecular chemistry simulation are projected to solve problems that are not solvable by even the largest of supercomputers due to the fact that a quantum machine inherently exhibits the properties of quantum particles such as molecules. Grover and QAOA are used to speed up large-scale optimization problems.

**Definition of Output Error.** As mentioned earlier, the output of a real quantum machine execution can be highly erroneous.

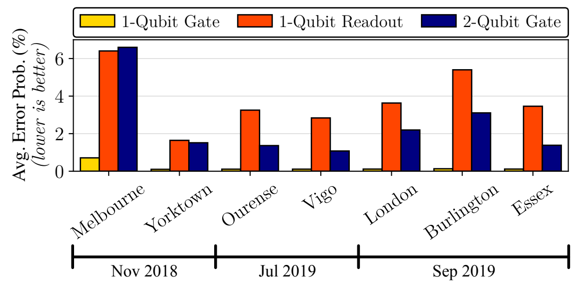


**Fig. 5.** Error probabilities of gate operations have improved from Melbourne to Essex (their online dates are shown at the bottom), 1-qubit readouts now the worst source of error.

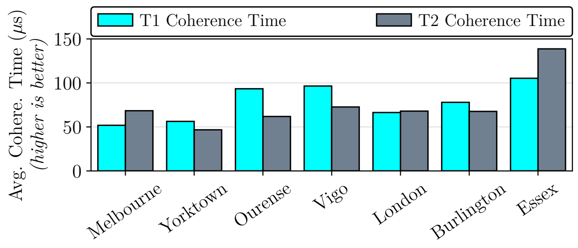


**Fig. 6.** T1 and T2 coherence times have improved considerably (more than doubled) from Melbourne to Essex.

For instance, if the real output of a 2-qubit program is $|00\rangle = 0.5$, $|01\rangle = 0.25$, $|10\rangle = 0.25$, and $|11\rangle = 0.0$. Then, the observed output can be $|00\rangle = 0.4$, $|01\rangle = 0.2$, $|10\rangle = 0.3$, and $|11\rangle = 0.1$. In this case, we define the *output error of a state* as the difference between the real and observed output of a state and the *overall output error* as the sum of these differences (divided by two to ensure the total error is less than 100%). In the example, the output error of each state is $e(|00\rangle) = 0.1$ or 10%, $e(|01\rangle) = 0.05$ or 5%, $e(|10\rangle) = 0.05$ or 5%, and $e(|11\rangle) = 0.1$ or 10%, and the overall output error is $(0.1 + 0.05 + 0.05 + 0.1)/2 = 0.15$ or 15%.

## IV. Benchmarking and Characterization of Cloud Quantum Computers

In this section, we report our lessons learned and their implications based on our extensive experimental benchmarking of the state-of-art IBM quantum computing cloud platforms.

### A. Fundamental Quantum Gate and Readout Operations: Errors and Speed

First, we study how the error probabilities of different quantum operations vary across different IBM QX machines and then, the speed of these quantum operations.

Fig. 5 shows the average error probabilities of 1-qubit gates, 1-qubit readouts, and 2-qubit gates on 7 IBM QX machines sorted according to their launch dates from Melbourne to Essex. For consistency, we have ensured that the error measurement and experimentation duration for all the machines are the same (over sixty days). However, the launch dates of individual machines are different.

We observe that the error probability of 1-qubit gates is significantly lower than the other two types of errors across all quantum machines, and it has improved over the generations. Essex's 1-qubit gates error probability is 0.001 or 0.1%, while on Melbourne, it is 0.7% (Essex is the latest quantum machine in IBM QX family, introduced in Sep'19). Note that meaningful quantum programs are expected to run on multiple qubits and perform long sequences of gates. Thus, even a 0.1% error probability for a single quantum operation can result in more than 50% error in program output [48, 55].

Fig. 5 shows that the error probability of 2-qubit gates has also improved considerably over generations, but it continues to be much higher than the error probability of 1-qubit gates ($\approx 12\times$ on average). A higher error probability for 2-qubit gates is expected since 2-qubit gates involve two qubits and have more microwave pulses, although the magnitude of difference high. Also, we note that 1-qubit readouts error probability is the highest on average among the three types of quantum operation errors ($\approx 1.5\times$ that of 2-qubit gates). It also has the least improvement as the technology matures.

One of the major reasons for readout errors being much higher than qubit gate errors is the change in the thermal environment and additional error-prone steps. Readout operations require probing the state of the qubit at the quantum chip level (15mK temperature) using microwave pulses, then, applying several layers of amplification of the output signal (at increasingly higher temperatures). This amplifies the noise in the signal, increasing the probability of misclassification of the output state. In contrast, quantum gates are applied and completed at the quantum chip level and remain in a relatively stable environment.

Next, we investigate the improvements in the coherence times. Recall that the T1 and T2 coherence times represent the amplitude and phase damping of a qubit's state to the ground state of $|0\rangle$. The longer these are, the more stable a qubit's state remains for a longer time. From Fig. 6, we observe that the T1 and T2 coherence times have improved from Melbourne to Essex (over $2\times$ improvement). Essex can run much longer quantum programs than can Melbourne without the qubit states decohering. We performed additional statistical tests using Spearman correlation (SC), which measures the correlation between two random variables (1 indicates a strong positive correlation, 0 indicates no correlation, and -1 indicates a strong negative correlation). The tests confirmed that the improvement in T1 coherence time is positively correlated with the improvement in 1-qubit (SC = 0.45) and 2-qubit gates (SC = 0.5), but negatively correlated with 1-qubit readouts (SC = -0.3). Improvement in coherence time is in conjunction with the improvement in 1-qubit and 2-qubit gate operation errors, but not readout errors.

**Observation 1.** *Over a span of one year, IBM's quantum machines have significantly improved in terms of gate error probabilities, and T1 and T2 coherence times. Readouts are now the biggest error source and have the largest impact on output reproducibility, thus requiring more focused research and development investments.*

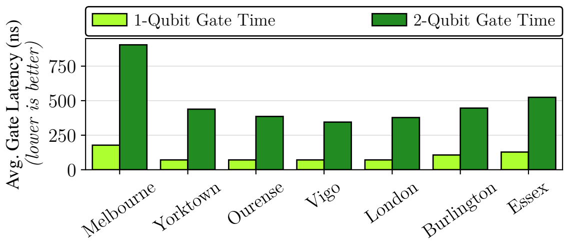


**Fig. 7.** The time to perform 1-qubit and 2-qubit gates has not decreased consistently in newer machines.

Next, we look at the latency of performing 1-qubit and 2-qubit gates in Fig. 7. Gate operation latency refers to the time it takes to apply a Microwave tone to change the state of a qubit. Note that a readout operation has a fixed latency across all qubits and machines; therefore, we do not study it here. On the other hand, gate operation latencies are calibrated twice daily to minimize gate operation error probabilities based on qubit conditions. Our experimental results indicate that 2-qubit gates take over $3\times$ the amount of time as 1-qubit gates across all the machines. For instance, on Melbourne, 1-qubit gates take 200ns on average, while 2-qubit operations take 900ns on average. This is of particular significance considering that Melbourne has a T1 coherence time of 50$\mu$s. This means that in the current state, a program can run a maximum of $\approx$55 error-free 2-qubit gates serially on Melbourne before the qubit state decoheres and produces highly erroneous output. Realistically, it can run much fewer operations because of errors introduced from applying the microwave pulses required for operations.

Further, when we compare Melbourne to Yorktown, we observe that the gate latencies have reduced. But, in general, they have not decreased on newer machines. There are two ways of improving the ability to perform more operations on quantum machines: (1) increase qubit coherence times, and (2) decrease quantum gate operation latencies. While coherence times have improved as we noted earlier, quantum gates operation times have not decreased. One reason for this is that gate latencies are calibrated for each qubit based on the daily conditions of the qubit, while coherence times are a direct result of the properties of the qubit. Since these properties depend on the manufacturing fidelity of the qubit, improvement in coherence times demonstrate more stable manufacturing technology, but not more stable environmental conditions.

**Observation 2.** *To be able to run longer programs with a higher number of operations, IBM QX has focused on improving coherence times as the first-order goal, instead of reducing the quantum gate operation latency. Our results reveal that, surprisingly, 2-qubit gates take $3\times$ longer than 1-qubit gates, but produce $12\times$ more erroneous results. As an implication, quantum compilers should focus on minimizing the number of 2-qubit gates if and when possible.*

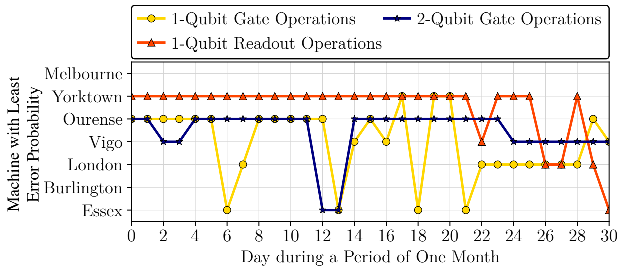


**Fig. 8.** Quantum machines which achieve the lowest operations errors change on temporal basis. In fact, different quantum machines are the best for different operations.

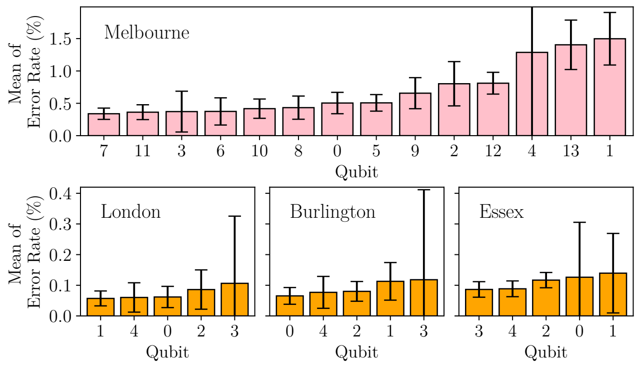


**Fig. 9.** Quantum operations have varying error probabilities on all machines from Melbourne (oldest) to Essex (newest).

Our experimental results revealed that different machines have different error probabilities and gate operation latencies on average. Next, we explore if these characteristics vary over time. We discovered that gate operation latencies remain relatively stable over time, but the error characteristics vary noticeably over time – unlike the classical computing domain.

Fig. 8 shows the quantum machine which achieves the best error probability on different days during one month for different types of operations (1-qubit gates, 1-qubit readouts, and 2-qubit gates). We observe that the machine which achieves the lowest of the three different types of operation error rates changes over time. Different quantum machines are the best for different operations. Note that in more that 85% of the samples the difference in error rates of the least erroneous machine and the second least erroneous machine is more than 5%. For instance, Yorktown most often has the lowest readout error probability, but this is not true on all days. Error probabilities of operations vary considerably temporally due to changes in the operational environment and qubit conditions, which cause variable errors in the program outputs too.

**Observation 3.** *Although quantum computing technology has been improving significantly; the optimal quantum machine for different types of quantum operations is not necessarily the latest one. In fact, the quantum machine that yields the least error probability for a given operation changes over time. Quantum application developers and resource managers should exploit time-varying heterogeneity among quantum machines to generate the least erroneous and most stable results, instead of relying on assumptions based on the chronological order of the machines.*

The fact that the best machine for different types of operations varies temporally indicates that error probabilities for different quantum qubits might be variable over time. To investigate this in detail, we look at the variability in error probabilities for different qubits on different machines.

Fig. 9 shows the mean error probability and the corresponding standard deviation (range indicators) for different qubits of multiple machines. First, we observe that some qubits might have lower error probability, but show higher variability – not a desirable characteristic for stability. For example, on Melbourne, qubit 4 has a lower average error probability than qubit 1 but a much higher variance. Qubit 2 and 12 have very similar error probabilities, but significantly different variances - making qubit 12 more desirable than qubit 2 as it is likely to provide less variable and more reproducible results. Second, this observed behavior is not limited to only older machines. However, it is also present on newer machines (e.g., on Essex, qubit 2, and 0 have similar mean error probability, but a large difference in the error variability).

Thus, when benchmarking the "quality" of a quantum operation, we should account for both factors: its fidelity (the average error probability) and its stability (considering the variance of error probability is critical since a highly variable operation error probability makes it difficult to make circuit mapping decisions for reproducibility). Therefore, we define a new metric: **"quality of an operation"**. The quality of an operation (1-qubit gate, 1-qubit readout, and 2-qubit gate) $= 1 - (\mu + \sigma)$, where $\mu$ is the mean error probability of the operation (0 if the error probability is 0% and 1 if the error probability is 100%) and $\sigma$ is the standard deviation of the operation error probability. The minimum value of $\mu + \sigma$ is 0. Therefore, the maximum quality of an operation is 1 (0 mean error probability and 0 variances). Note that previous works have proposed the metric of "quantum volume" to evaluate the performance of a quantum computer using randomly-generated circuits which takes into account the number of qubits on the computer, the error rate of the worst qubit operation, and the qubit connectivity [13]. However, the metric is used for the evaluation of a quantum computer as a whole, and therefore does not take into account difference among errors of different qubits, nor does it take into account the error variance of a qubit.

Fig. 10, 11, and 12 show the quality of individual 1-qubit gates, 1-qubit readouts, and 2-qubit gates, respectively. First, we observe that the quality of 1-qubit gates is very high (very close to 1) in general and has improved from Melbourne to Essex. As expected, the quality of 1-qubit readouts is lower in general and has not improved much from Melbourne to Essex. The quality of 2-qubit gates is lower than 1-qubit gates

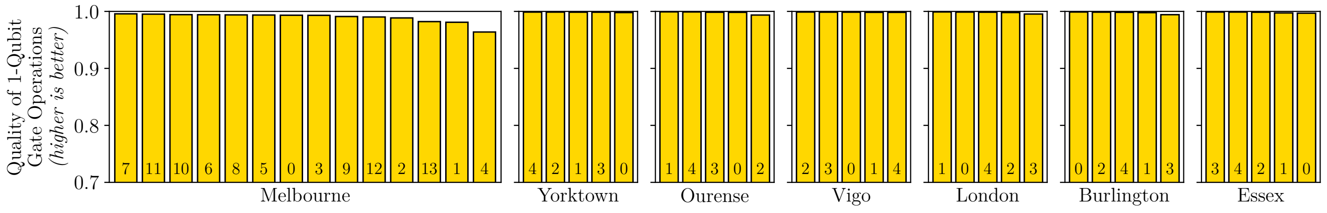


**Fig. 10.** Quality of all 1-qubit gates on all IBM QX quantum machines (sorted by the quality). The numbers inside the bars indicate the IDs of the 1-qubit gates. The quality of 1-qubit gates is very high in general and has improved from Melbourne to Essex.

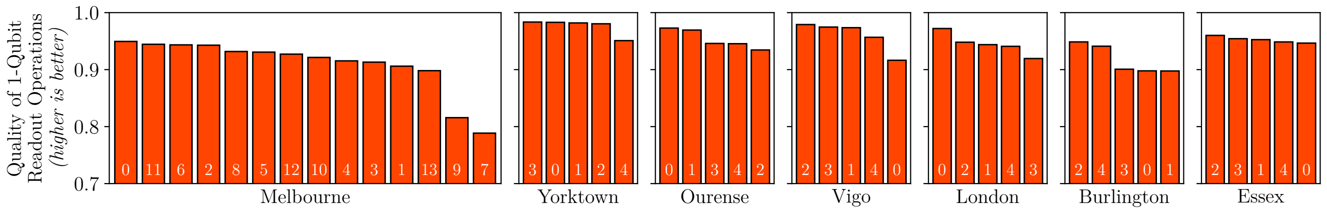


**Fig. 11.** Quality of all 1-qubit readouts on all IBM QX quantum machines (sorted by the quality). The numbers inside the bars indicate the IDs of the 1-qubit readouts. The quality of 1-qubit readouts is low in general and has not improved considerably from Melbourne to Essex.

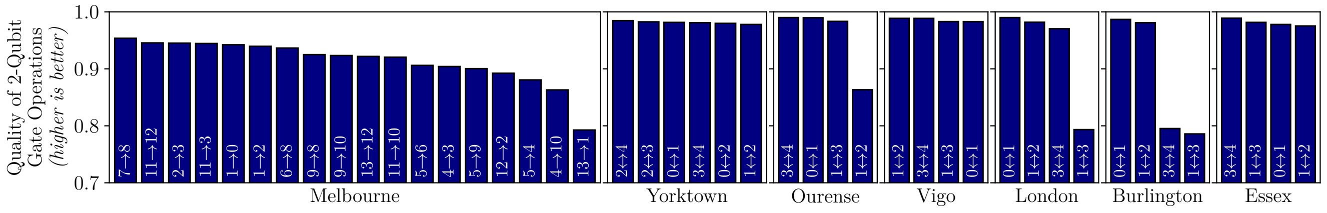


**Fig. 12.** Quality of all 2-qubit gates on all IBM QX quantum machines (sorted by the quality). The numbers inside the bars indicate the IDs of the 2-qubit gates (the arrow indicates the direction from control to target qubit – bidirectional arrow indicates that the gate can be applied in both directions). The quality of 2-qubit gates is lower than 1-qubit gates in general and has improved from Melbourne to Essex.

and higher than 1-qubit readouts in general and has improved from Melbourne to Essex. Second, the quality of qubits within the same machine varies significantly (e.g., from 0.80 to 0.95) even on newer machines (e.g., Burlington).

**Observation 4.** *When assessing the quality of a qubit operation, both its mean error probability (its fidelity) and the variance of its error probability (its stability) should be taken into account. The quality of 1-qubit gates is high and has improved considerably with newer quantum machines. The quality of 1-qubit readouts is much worse and has improved the least in newer machines. Interestingly, the quality of qubits within the same machine varies significantly, suggesting a need for careful qubit mapping for quantum programs. Currently, the state-of-the-art research proposals for error-aware circuit mappings [3, 32, 51, 53] only take the average error rate or the most recently reported error rate for different qubits into considerations, not their variance.*

## B. Errors in Quantum Program Execution

In the previous section, we studied the characteristics of quantum operations on different quantum machines. Next, *we investigate how these characteristics affect the overall execution of different real-world quantum programs, in terms of the overall observed error in the program output and execution time of the program.*

First, we calculate the overall observed error in the program output for different quantum programs on different quantum machines. From Fig. 13, we observe that there is no single quantum machine that is the best platform across all quantum programs: the newest is not always the best.

For example, on average, Yorktown is the best machine for CHEM, while Vigo is the best machine for BV, DJ, QFT, QPE, SIMON, and 3SAT, Burlington machine is the best for QAOA, and Ourense machine is the best for GROVER. This is because each program performs different types and numbers of quantum operations, and each machine has a different output error for different types of operations. The CHEM program has the lowest output error on the Yorktown machine because it has gates between all combinations of pairs of qubits. From Fig. 2, Yorktown has the most number of direct 2-qubit connections, minimizing the need to swap qubits for 2-qubit operations, thus reducing the output error. On the other hand, the GROVER program has the highest number of 2-qubit gates

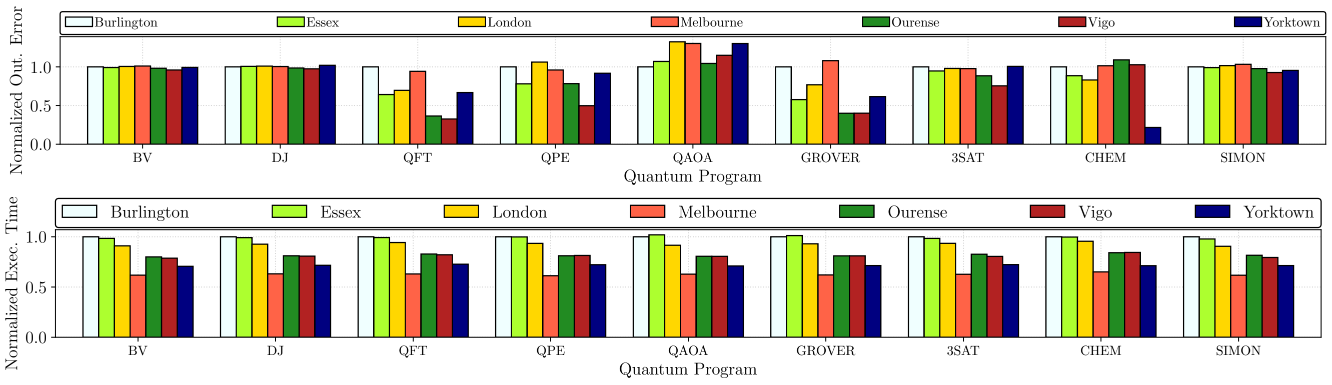


**Fig. 13.** The quantum machine which performs the best on average varies from program to program. But, the ranking of quantum machines in terms of execution time remains consistent for all programs.

and, therefore, performs the best on Ourense. Recall from Fig. 8, Ourense most often has the lowest 2-qubit gate errors.

This is particularly notable since it shows that end-users should not trivially assume that the latest quantum machine is likely to produce a minimal error for their quantum program. This is against intuition since quantum computing platforms are expected to improve over generations and produce lesser errors in the program output. While this expectation is valid over the longer-term as we move further along in the NISQ era, our results indicate that the choice among contemporary quantum machines that are operational at the same time is not straight forward.

Along with minimizing errors in the program output for quantum programs, quantum computing users are also interested in running their programs fast. To explore this trade-off, next, we investigate the execution time characteristics.

In Fig. 13, we observe that unlike overall program output error characteristics, a single quantum machine (Melbourne) is optimal for all quantum programs in terms of execution times. Interestingly, while the fastest, Melbourne has one of the highest output errors for all programs. The finding implies that the programmers or resource managers cannot use "execution time performance" as the proxy for the best machine choice. However, a quantum program that is inherently not sensitive to error in the program output or has only one dominant output can exploit this trade-off to have a preferred mapping machine based on execution time alone (Melbourne).

Second, we observe that although the variation among machines in terms of program output error is significant across different programs (Fig. 13), the same is not valid for execution time – the ranking of quantum machines in terms of execution times remains the same across all programs (Fig. 13). This implies that minimizing program output error should be the primary objective for a given program since the ranking of machines in terms of error rate varies across programs, but not for execution time.

**Observation 5.** *Different programs have their lowest output error on different quantum machines based on the composition of their gate and readout operations. Moreover, the execution times of programs also vary significantly depending on the machine, and the machine which has the lowest execution times does not necessarily have the lowest output errors. The quantum machine to run the program should not be selected based on execution time as the output reproducibility might be compromised.*

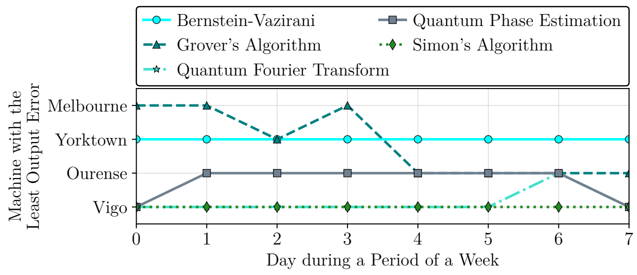
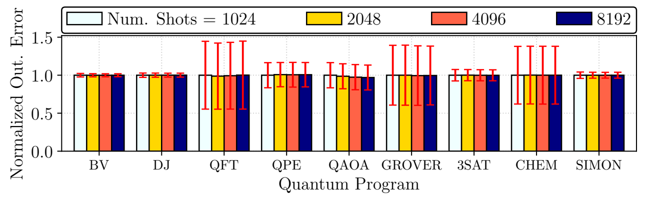


**Fig. 14.** The best quantum machine in terms of lowest output errors varies temporally for different quantum programs.

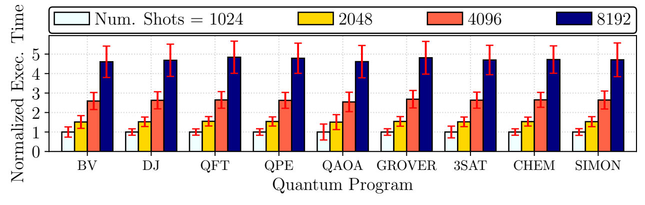
Now that we have answered the question "which quantum machine is the best for different quantum programs *on average*?", we ask whether the selection of the best quantum machine for a given program is temporally varying. To evaluate this, we conducted experiments with a subset of the quantum programs on four quantum machines for seven days during which the machines were available. These programs include the BV, GROVER, QFT, QPE, and SIMON programs.

Surprisingly, our results (Fig. 14) reveal that the best quantum machine for each quantum circuit is not constant - it varies over different days for different programs. For instance, while the Bernstein-Vazirani (BV) program always achieves the lowest output error when run on Yorktown in this case, Grover's program can have the lowest output error on either Melbourne or Yorktown or Ourense, depending on the day. This result implies the machine which performs the best on average is not necessarily always the best one.

**Observation 6.** *While different programs have different machines that perform the best on average, the best machine varies temporally on as frequently as daily basis. Quantum*



**(a)**



**(b)**

**Fig. 15.** (a) Increasing the number of shots does not decrease the output error of the programs. (b) On the other hand, increasing the number of shots does have a significant negative effect on the execution time across all programs.

*application schedulers should leverage temporal variability among quantum machines to select the most optimal machine for different programs. Unlike classical computing, the selection process needs to be repeated frequently over time to reduce the output error optimally. As a good benchmarking practice, future studies should specify research results claiming performance improvements by running experiments on multiple machines and over multiple spans of days to account for temporal and spatial variability.*

## C. Impact of User-configurable Options on the Quantum Program Output Error

While it is crucial to understand how quantum machines affect program output errors and execution times, it is also essential to understand how user-configurable parameters available with Qiskit affect these characteristics on IBM QX platforms. In particular, we study the impact of the two most popular options: "number of shots" and "compiler optimization level".

**Number of shots.** First, we study how the number of shots affects the output errors and execution times of different programs in Fig. 15. The bars show the mean output error across all runs across different computers while the range indicators show the standard deviation of the error. Recall that the number of shots is the number of times the program circuit is run to get the probability of output states. One can reasonably hypothesize that increasing the number of shots would reduce the error as more samples tend to provide better estimates of real probabilities. We vary the number of shots from 1024 to the maximum limit of 8192.

Surprisingly, increasing the number of shots does not reduce the output error as much as one might anticipate. Across all programs, when the number of shots is increased by 8× from 1024 to 8192, there is a limited improvement in the observed output error. For QAOA and GROVER, the output error decreases slightly (less than 2%) as the number of shots is increased to 8192. However, there is no noticeable improvement for other programs. The reason for this is that the output error is not a result of a lack of enough sampling but a result of the errors inherent to a quantum machine. Merely collecting more samples by running more shots cannot eliminate these operation errors. Increasing the number of shots does not reduce the variability in output error among multiple runs even when multiple runs are conducted with the same number of shots. We note that the variability in output error is particularly high for QFT, GROVER, and CHEM because of the high variability in their output errors across different machines, as shown in Fig. 13.

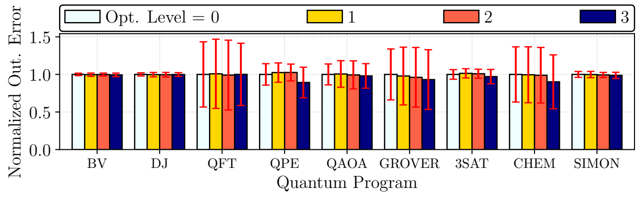
However, as expected, increasing the number of shots increases the execution time of the programs proportionally to the number of shots, as shown in Fig. 15(b).

**Observation 7.** *Running more than 1024 shots only increases the execution time across different programs, while not reducing the output error or error variability. This finding implies that instead of increasing the number of "shots" during one run, programmers should instead try to do multiple runs over time on different machines to exploit the temporal and spatial variations in error characteristics. This practice can potentially lead to better program output estimates and confidence intervals.*

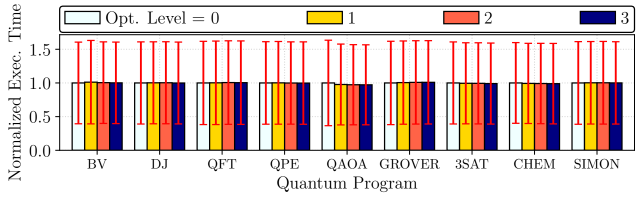
**Compiler Optimization.** The compiler optimization level determines the types of optimizations that are applied when the quantum program is mapped to a quantum circuit on real quantum hardware. *The main objective behind applying these optimizations is to reduce the number of gate operations (which would reduce operations-specific errors) and the depth of the circuit. The circuit depth corresponds to the most number of serially run operations in the circuit. In essence, a shorter depth decreases the likelihood of qubits from losing their coherent states before the circuit is completed.* As described in Sec. III, there are four optimization levels: 0, 1, 2, and 3, each performing progressively more number of operations and circuit optimizations.

Fig. 16 shows the impact of different optimization levels on the output errors and execution times of different quantum programs. As expected, the optimization levels have different impacts on different programs in terms of output errors. For programs such as QPE, QAOA, GROVER, and CHEM, increasing the optimization level results in a significant decrease in the output errors. For example, the output error is reduced by 12% when optimization level 3 is used for CHEM as opposed to using optimization level 0. However, for the remaining programs, there is little to no impact of applying higher optimization levels on the output errors.

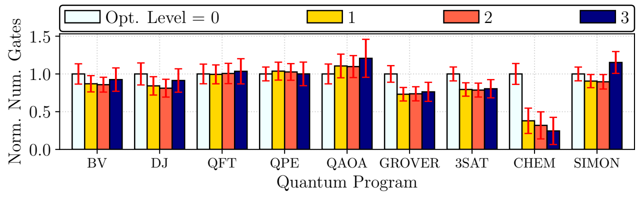
*Interestingly, we observe that higher optimization levels also tend to introduce more variability to the output errors.* For example, QPE has a standard deviation of 10% in the output error with optimization level 0 but has a standard deviation



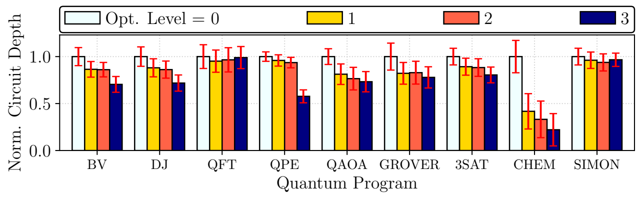
(a)



(b)



(c)



(d)

**Fig. 16.** (a) Increasing the optimization level reduces the output errors slightly for some programs, but it does not reduce the variability in output errors. (b) Increasing the optimization level neither improves the mean execution time, nor does it reduce the variability in execution time. In fact, increasing the optimization level reduces the (c) number of gates and (d) circuit depth for most programs, but the amount of reduction varies as per the program.

of 18% in the output error when the optimization level is 3, even though the mean output error at optimization level is lower by 17%. This is because the optimizations applied to quantum circuits have a certain level of stochasticity as they depend on the characteristics of the operations. Therefore, applying more optimization leads to more randomness in the results and higher variability in the output errors, which in turn makes the output less reproducible. Thus, higher optimization levels need to be considered carefully due to potential higher variability, especially given that they do not improve the output error across all programs.

Next, when studying the impact of the optimization levels on the execution time, we observe that the execution times do not significantly get affected for any of the programs. This is surprising given the fact that higher optimization levels should

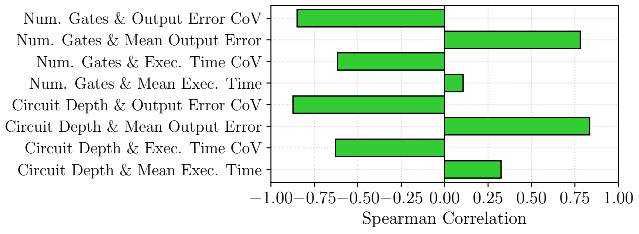


**Fig. 17.** Circuit depth and number of gates are positively correlated with output error, but not execution time. On the contrary, both are negatively correlated with variability in execution time and output error.

reduce the number of operations and circuit depth, which in turn is expected to reduce the execution times of the circuits. However, since this is not the case, we dig deeper into how the selection of different optimization levels affects the number of gates and circuit depth.

Fig. 16(c) shows the number of gates and Fig. 16(d) shows the circuit depth for different optimization levels for all the programs. The first observation is that increasing the optimization level does not necessarily reduce the number of gates. In fact, for programs such as BV, DJ, QFT, QAOA, and SIMON, the number of gates can be up to 20% higher on average with optimization level 3 than with optimization level 1. *This points to the fact that reducing the number of gates is not the primary objective of compiler optimization. In fact, it has another more important objective for which it is willing to compromise in terms of the number of gates. This primary objective is to reduce the circuit depth, as discussed next.*

Fig. 16(d) shows that circuit depth is significantly reduced by increasing the optimization level. For programs such as CHEM, the reduction in circuit depth is over 70% with optimization level 3 as compared to optimization level 0. Notice that all programs which have some improvement in output errors in Fig. 16(a), such as QPE, QAOA, GROVER, 3SAT, and CHEM, also have a vast improvement in their circuit depths at higher optimization levels. This supports that the strategy of optimizing the circuit depth as the primary objective at the cost of increasing the number of gates is useful for some programs. However, it is not helpful for other programs, as their output errors do not improve. Even for programs with reduced output errors, there is no improvement in execution times.

To study this further, in Fig. 17, we calculated the Spearman correlation between the number of gates & circuit depth and output error & its variability across all programs. The correlation is strong between the number of gates/circuit depth and the mean output error. Reducing them both can help reduce the mean output error. However, current optimization applied by the Qiskit compiler reduces the circuit depth at the risk of increasing the number of gates as we saw earlier. Therefore, overall, the reduction in mean output error is not realized. Surprisingly, both the number of gates and circuit depth is strongly negatively correlated with the coefficient of variation or CoV (standard deviation divided by the mean) of

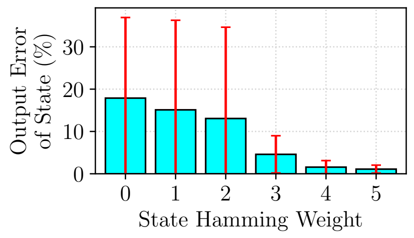


**Fig. 18.** States with higher hamming weight tend to have lower output errors and lower variability in output errors.

output errors. This demonstrates that reducing the circuit depth and the number of gates of a circuit increases the variability of the output error, introducing undesired uncertainty to the program output.

**Observation 8.** *Selecting higher optimization levels does not necessarily improve the output error or the execution time of the programs. This is because the primary metric that the compiler optimizes is circuit depth, even if it is at the cost of increasing the number of gate operations. Our results indicate that reducing the circuit depth and the number of gates of a circuit increases the variability of the output error. Thus, incautiously selecting the highest optimization level can also have undesired side-effects, such as an increase in the variability of output errors and even execution time in some cases. Hence, choosing an effective optimization level may require careful tuning, as it depends heavily on the program and the quantum computer. The chosen setting should be specified when reporting results.*

### D. Errors Affected by the Characteristics of Individual Output States of Quantum Programs

Previously, we studied the impact of errors on the final output of different quantum programs. Recall that the observed error in the final output of a program is simply an artifact of individual output states producing incorrect probabilities (as defined in Sec. III). For example, a simple 2-qubit quantum program has four output states: $|00\rangle$, $|01\rangle$, $|10\rangle$ and $|11\rangle$. A $n$-qubit quantum output can have $2^n$ output states, each with individual probabilities. Because of errors in qubit operations and qubit decoherence, these output states themselves may produce different output probabilities than expected (or correct) output probabilities. These deviations from the correct output probabilities of individual states add up and manifest as the observed error in the final output of a program.

*We investigate if certain output states are more prone to higher error than others*. We *hypothesize* that output states with higher hamming weight (i.e., the number of 1s in the output state) have a higher error in output probability. For example, output state $|11\rangle$ should be more error-prone than $|00\rangle$ or $|10\rangle$. The intuition behind this hypothesis is that a higher number of 1s is likely to make the output state decohere to the ground state more quickly (e.g.. state $|11\rangle$ can decohere into state $|10\rangle$, $|01\rangle$, or $|00\rangle$). Notably, this hypothesis has been held by quantum computing researchers and exploited by Tannu et al. [54] to actively correct the readout errors of qubits with output state $|1\rangle$ by applying the NOT gate when the qubit value is expected to be $|1\rangle$ to reduce the impact of decoherence. We test the validity of this hypothesis using the experimental results collected from all the runs performed on a wide variety of machines and quantum programs.

Our results in Fig. 18 reveal that, contrary to our hypothesis, the output states with higher hamming weights have lower overall error (and lower error variability too). This is because the output states with higher hamming weights tend to decohere and simultaneously, "transfer" errors to states with lower hamming weights. For example, if state $|11\rangle$ has 10% error (say, the observed probability is 0.1 less than its expected probability), that error must be transferred to other states who would be observing increased probabilities from their expected probability (recall that the probabilities across all output states add up to one). These outputs states generally tend to be states with lower hamming weights due to qubit decoherence. Thus, the output state that has zero hamming weight accumulates errors from all other states with hamming weights higher than it (Fig. 18. The accumulation of errors, therefore, tends to decrease as hamming weight increases, which results in states with higher hamming weight experiencing a lower output error than states with lower hamming weight.

**Observation 9.** *Output states with lower hamming weights tend to experience higher output errors due to the accumulation of errors from states with higher hamming weight. This finding is in contrast with conventional wisdom held in the quantum computing architecture community [54] and shows that efforts targeting mitigation of errors only on higher hamming weight states could be counterproductive. Instead, we should prioritize error mitigation for output states in the reverse order of their hamming weight, after executing the quantum program. A useful implication for quantum programmers is to map the important states of the program output carefully to the states with higher hamming weights.*

## V. Scope of Our Findings

All of our experiments are conducted on the IBM cloud quantum computers. These systems represent a diverse range of characteristics in terms of number of qubits, topology, and qubit error and coherence characteristics. All of these systems use IBM's superconducting qubit technology. Superconducting qubit technology is promising due to its scalability and has garnered wide research interest from industry and academia, as described in the next section (Sec. VI). However, we do not claim that our findings are applicable as-is to other types of quantum computing technologies such as trapped-ion qubits and photon-based qubits. Secondly, as quantum computing technologies mature, we expect that larger-scale HPC applications will be executed on quantum computers with 100s of qubits, and further insights will be derived.

## VI. Related Work

In this section, we discuss relevant NISQ computing works.

**Impractical Quantum Error Correction (QEC).** Previous works have proposed qubit ECC codes that rely on heavy computational requirements and high spatial overhead. They are therefore largely unsuitable for present NISQ computing devices [4, 7, 23, 30, 52, 56, 58]. As these methods are not possible to apply on NISQ technology, our analysis and characterization of errors in current quantum operations can help researchers develop error mitigation and reproducibility techniques for existing quantum machines.

**Improving the Stability of Qubits.** Several feedback-control techniques have been proposed to realize stable quantum systems [57, 61]. However, due to weak state-measurement capabilities, feedback control undergoes state collapse for most qubit technologies [47]. Superconducting qubit technology used by IBM is the most promising [26, 28]. However, the current state-of-the-art IBM quantum machines have not been able to achieve anywhere close to stable qubits, thus, requiring qubit-quality-aware improvement in experimental methodologies as proposed by our work.

**Error Rates Minimization.** With the introduction of quantum programming language and frameworks such as IBM Qiskit [2], Microsoft Q# [1], Google Cirq [21], and others [12, 27], many recent works have developed strategies to mitigate the effects of operation errors in quantum circuits via online and offline optimizations [3, 8, 14, 18, 24, 32, 33, 39–41, 50, 51, 51, 54, 55, 59, 62, 63]. These include using multiple circuit maps to minimize output error [51], using machine learning and adaptive control techniques for estimation of future qubit state based on past outcomes [37, 44], using post-execution error correction to find correct program output [46], using OpenPulse to classify the output states accurately [45]. Future works along these directions would benefit from incorporating our in-depth analysis of NISQ qubits, operations, and errors.

## VII. Conclusion

In this paper, we measured, collected, and provided a data-driven characterization of seven quantum machines available on IBM's QX cloud. We discovered the rapid improvement in T1 and T2 coherence times and gate error probabilities, but the lack of improvement in readout errors and gate latencies. We provided a new method to assess the quality of quantum operations as technology progresses. We analyzed factors that affect the reproducibility of HPC quantum programs. Our findings and open-source dataset can help drive experimental methodology on future quantum machines. Our study is open-sourced at `https://doi.org/10.5281/zenodo.3957894`.

**Acknowledgment.** We are thankful to anonymous reviewers to provided thoughtful feedback despite the pandemic challenges. We are thankful for the support from Northeastern University, NSF Award 1910601, and the Massachusetts Green High Performance Computing Center (MGHPCC) facility. We acknowledge use of the IBM Q for this work. The views expressed are those of the authors and do not reflect the official policy or position of IBM or the IBM Q team.

## References

[1] Talha Ahmed. *Q#: A Quantum Programming Language by Microsoft*. PhD thesis, Imperial College London, 2018.

[2] G Aleksandrowicz, T Alexander, P Barkoutsos, L Bello, Y Ben-Haim, D Bucher, et al. Qiskit: An Open-source Framework for Quantum Computing.(2019).

[3] Abdullah Ash-Saki, Mahabubul Alam, and Swaroop Ghosh. QURE: Qubit Re-allocation in Noisy Intermediate-Scale Quantum Computers. In *Proceedings of the 56th Annual Design Automation Conference 2019*, page 141. ACM, 2019.

[4] Charles H Bennett, David P DiVincenzo, John A Smolin, and William K Wootters. Mixed-State Entanglement and Quantum Error Correction. *Physical Review A*, 54(5), 1996.

[5] Ethan Bernstein and Umesh Vazirani. Quantum Complexity Theory. *SIAM Journal on computing*, 26(5):1411–1473, 1997.

[6] Sergey Bravyi, Graeme Smith, and John A Smolin. Trading Classical and Quantum Computational Resources. *Physical Review X*, 6(2):021043, 2016.

[7] Jonathan J Burnett, Andreas Bengtsson, Marco Scigliuzzo, David Niepce, Marina Kudra, Per Delsing, and Jonas Bylander. Decoherence Benchmarking of Superconducting Qubits. *npj Quantum Information*, 5(1), 2019.

[8] Anastasiia Butko, George Michelogiannakis, Samuel Williams, Costin Iancu, David Donofrio, John Shalf, Jonathan Carter, and Irfan Siddiqi. Understanding quantum control processor capabilities and limitations through circuit characterization. *arXiv preprint arXiv:1909.11719*, 2019.

[9] J Ignacio Cirac and Peter Zoller. Goals and Opportunities in Quantum Simulation. *Nature Physics*, 8(4):264, 2012.

[10] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca. Quantum Algorithms Revisited. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 454(1969):339–354, Jan 1998.

[11] David Collins, KW Kim, and WC Holton. Deutsch-Jozsa Algorithm as a Test of Quantum Computation. *Physical Review A*, 58(3):R1633, 1998.

[12] Rigetti Computing. Pyquil documentation. *URL http://pyquil. readthedocs. io/en/latest*, 2019.

[13] Andrew W Cross, Lev S Bishop, Sarah Sheldon, Paul D Nation, and Jay M Gambetta. Validating quantum computers using randomized model circuits. *Physical Review A*, 100(3):032328, 2019.

[14] Poulami Das, Swamit S Tannu, Prashant J Nair, and Moinuddin Qureshi. A Case for Multi-Programming Quantum Computers. In *Proceedings of the 52nd Annual*

*IEEE/ACM International Symposium on Microarchitecture*, pages 291–303. ACM, 2019.

[15] Michel H Devoret and Robert J Schoelkopf. Superconducting Circuits for Quantum Information: An Outlook. *Science*, 339(6124), 2013.

[16] Miroslav Dobšíček, Göran Johansson, Vitaly Shumeiko, and Göran Wendin. Arbitrary Accuracy Iterative Quantum Phase Estimation Algorithm using a Single Ancillary Qubit: A Two-Qubit Benchmark. *Physical Review A*, 76(3):030306, 2007.

[17] Edward Farhi and Aram W Harrow. Quantum Supremacy through the Quantum Approximate Optimization Algorithm. *arXiv preprint arXiv:1602.07674*, 2016.

[18] Pranav Gokhale, Yongshan Ding, Thomas Propson, Christopher Winkler, Nelson Leung, Yunong Shi, David I Schuster, Henry Hoffmann, and Frederic T Chong. Partial Compilation of Variational Algorithms for Noisy Intermediate-Scale Quantum Machines. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 266–278. ACM, 2019.

[19] Lov K Grover. A Fast Quantum Mechanical Algorithm for Database Search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996.

[20] Kathleen E Hamilton, Tyler Kharazi, Titus Morris, Alexander J McCaskey, Ryan S Bennink, and Raphael C Pooser. Scalable quantum processor noise characterization. *arXiv preprint arXiv:2006.01805*, 2020.

[21] Andrew Hancock, Austin Garcia, Jacob Shedenhelm, Jordan Cowen, and Calista Carey. Cirq: A python framework for creating, editing, and invoking quantum circuits.

[22] Mohamed W Hassan, Scott Pakin, and Wu-chun Feng. C to D-Wave: A High-level C Compilation Framework for Quantum Annealers. In *2019 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–8. IEEE, 2019.

[23] Eric Huang, Andrew C Doherty, and Steven Flammia. Performance of Quantum Error Correction with Coherent Errors. *Physical Review A*, 99(2), 2019.

[24] Yipeng Huang and Margaret Martonosi. Statistical Assertions for Validating Patterns and Finding Bugs in Quantum Programs. In *Proceedings of the 46th International Symposium on Computer Architecture*, pages 541–553. ACM, 2019.

[25] Lawrence Ioannou. Continuous-Time Quantum Algorithms: Searching and Adiabatic Computation. Master's thesis, University of Waterloo, 2002.

[26] Andrew J Kerman. Superconducting Qubit Circuit Emulation of a Vector Spin-1/2. *New Journal of Physics*, 21(7), 2019.

[27] Nathan Killoran, Josh Izaac, Nicolás Quesada, Ville Bergholm, Matthew Amy, and Christian Weedbrook. Strawberry fields: A software platform for photonic quantum computing. *Quantum*, 3:129, 2019.

[28] Morten Kjaergaard, Mollie E Schwartz, Jochen Braumüller, Philip Krantz, Joel I-Jan Wang, Simon Gustavsson, and William D Oliver. Superconducting Qubits: Current State of Play. *arXiv preprint arXiv:1905.13641*, 2019.

[29] Pascal Koiran, Vincent Nesme, and Natacha Portier. A Quantum Lower Bound for the Query Complexity of Simon's Problem. In *International Colloquium on Automata, Languages, and Programming*, pages 1287–1298. Springer, 2005.

[30] David Layden, Sisi Zhou, Paola Cappellaro, and Liang Jiang. Ancilla-Free Quantum Error Correction Codes for Quantum Metrology. *Physical review letters*, 122(4), 2019.

[31] Ang Li and Sriram Krishnamoorthy. Qasmbench: A low-level qasm benchmark suite for nisq evaluation and simulation. *arXiv preprint arXiv:2005.13018*, 2020.

[32] Gushu Li, Yufei Ding, and Yuan Xie. Tackling the Qubit Mapping Problem for NISQ-Era Quantum Devices. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 1001–1014. ACM, 2019.

[33] Gushu Li, Yufei Ding, and Yuan Xie. Towards efficient superconducting quantum processor architecture design. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 1031–1045, 2020.

[34] Ji Liu, Gregory T Byrd, and Huiyang Zhou. Quantum circuits for dynamic runtime assertions in quantum computation. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 1017–1030, 2020.

[35] Piotr R Luszczek, David H Bailey, Jack J Dongarra, Jeremy Kepner, Robert F Lucas, Rolf Rabenseifner, and Daisuke Takahashi. The hpc challenge (hpcc) benchmark suite. In *Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, volume 213, pages 1188455–1188677. Citeseer, 2006.

[36] Margaret Martonosi and Martin Roetteler. Next Steps in Quantum Computing: Computer Science's Role. *arXiv preprint arXiv:1903.10541*, 2019.

[37] Mavadia et al. Prediction and Real-Time Compensation of Qubit Decoherence via Machine Learning. *Nature communications*, 8, 2017.

[38] Jarrod McClean. Variational Quantum Eigensolver: How to Use Any Quantum Device in Your Lab to Perform Quantum Simulation. In *APS Meeting Abstracts*, 2015.

[39] Prakash Murali, Jonathan M Baker, Ali Javadi-Abhari, Frederic T Chong, and Margaret Martonosi. Noise-Adaptive Compiler Mappings for Noisy Intermediate-Scale Quantum Computers. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 1015–1029. ACM, 2019.

[40] Prakash Murali, David C McKay, Margaret Martonosi, and Ali Javadi-Abhari. Software mitigation of crosstalk on noisy intermediate-scale quantum computers. pages 1003–1016, 2020.

[41] Daniel C Murphy and Kenneth R Brown. Controlling Error Orientation to Improve Quantum Algorithm Success Rates. *Physical Review A*, 99(3):032318, 2019.

[42] Scott Pakin. A Quantum Macro Assembler. In *2016 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–8. IEEE, 2016.

[43] Scott Pakin and Steven P Reinhardt. A Survey of Programming Tools for D-Wave Quantum-Annealing Processors. In *International Conference on High Performance Computing*, pages 103–122. Springer, 2018.

[44] Tirthak Patel, Baolin Li, Rohan Basu Roy, and Devesh Tiwari. UREQA: Leveraging Operation-Aware Error Rates for Effective Quantum Circuit Mapping on NISQ-Era Quantum Computers. In *2020 {USENIX} Annual Technical Conference ({USENIX}{ATC} 20)*, pages 705–711, 2020.

[45] Tirthak Patel and Devesh Tiwari. DisQ: A Novel Quantum Output State Classification Method on IBM Quantum Computers using OpenPulse. In *ICCAD*, 2020.

[46] Tirthak Patel and Devesh Tiwari. VERITAS: Accurately Estimating the Correct Output on Noisy Intermediate-Scale Quantum Computers. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2020.

[47] LA Poyneer. Control of Superconducting Quantum Circuits (17-ERD-006). Technical report, Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), 2019.

[48] John Preskill. Quantum Computing in the NISQ Era and Beyond. *Quantum*, 2:79, 2018.

[49] David F Richards, Omar Aaziz, Jeanine Cook, Hal Finkel, Brian Homerding, Peter McCorquodale, Tiffany Mintz, Shirley Moore, Abhinacv Bhatele, and Robert Pavel. Fy18 proxy app suite release. milestone report for the ecp proxy app project. Technical report, Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), 2018.

[50] Yunong Shi, Nelson Leung, Pranav Gokhale, Zane Rossi, David I Schuster, Henry Hoffmann, and Frederic T Chong. Optimized Compilation of Aggregated Instructions for Realistic Quantum Computers. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 1031–1044. ACM, 2019.

[51] Kaitlin N Smith and Mitchell A Thornton. A Quantum Computational Compiler and Design Tool for Technology-Specific Targets. In *Proceedings of the 46th International Symposium on Computer Architecture*, pages 579–588. ACM, 2019.

[52] Luyan Sun, Ling Hu, Yuwei Ma, Weizhou Cai, Xianghao Mu, Yuan Xu, Wang Weiting, Yukai Wu, Haiyan Wang, Yipu Song, et al. Experimental Quantum Error Correction with Binomial Bosonic Codes. In *APS Meeting Abstracts*, 2019.

[53] Swamit S Tannu and Moinuddin Qureshi. Ensemble of Diverse Mappings: Improving Reliability of Quantum Computers by Orchestrating Dissimilar Mistakes. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 253–265. ACM, 2019.

[54] Swamit S Tannu and Moinuddin K Qureshi. Mitigating Measurement Errors in Quantum Computers by Exploiting State-Dependent Bias. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 279–290. ACM, 2019.

[55] Swamit S Tannu and Moinuddin K Qureshi. Not All Aubits are Created Equal: A Case for Variability-Aware Policies for NISQ-Era Quantum Computers. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 987–999. ACM, 2019.

[56] Barbara Terhal, Leonid Pryadko, Daniel Weigand, Yang Wang, Hamed Asasi, and Christophe Vuillot. Scalable Quantum Error Correction with the Bosonic GKP Code. In *APS Meeting Abstracts*, 2019.

[57] Lorenza Viola and Seth Lloyd. Dynamical Suppression of Decoherence in Two-State Quantum Systems. *Phys. Rev. A*, 58, Oct 1998.

[58] Christophe Vuillot, Hamed Asasi, Yang Wang, Leonid P Pryadko, and Barbara M Terhal. Quantum Error Correction with the Toric Gottesman-Kitaev-Preskill Code. *Physical Review A*, 99(3), 2019.

[59] Robert Wille, Lukas Burgholzer, and Alwin Zulehner. Mapping Quantum Circuits to IBM QX Architectures Using the Minimal Number of SWAP and H Operations. In *Proceedings of the 56th Annual Design Automation Conference 2019*, page 142. ACM, 2019.

[60] Ellis Wilson, Sudhakar Singh, and Frank Mueller. Just-in-time quantum circuit transpilation reduces noise. *arXiv preprint arXiv:2005.12820*, 2020.

[61] H. M. Wiseman and A. C. Doherty. Optimal Unravellings for Feedback Control in Linear Quantum Systems. *Phys. Rev. Lett.*, 94, Feb 2005.

[62] Alwin Zulehner, Alexandru Paler, and Robert Wille. An Efficient Methodology for Mapping Quantum Circuits to the IBM QX Architectures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2018.

[63] Alwin Zulehner and Robert Wille. Compiling SU (4) Quantum Circuits to IBM QX Architectures. In *Proceedings of the 24th Asia and South Pacific Design Automation Conference*, pages 185–190. ACM, 2019.