Efficient Programming of Near- and Long-Term Quantum Computers

Ali Javadi-Abhari
IBM Research
IBM Q Experience

80K Unique Users

3 Million Experiments Run

70 Scientific Papers

Quantum Goes Global

The IBM Quantum Experience has attracted an enthusiastic community of users. Here's a sampling of the activities from experiments and courses to plenary sessions – built around our 5-qubit machine.
From Quantum Experience to Quantum Programs

QISKit

Build

Compile

Python Interface

Translate & Optimize

Quantum Developers

github.com/QISKit/qiskit-sdk-py/

API

Real Devices

Simulators

Laboratory
QISKit Architecture

- Libraries
- OpenQASM
- dag Entangler
- Decompose to supported gates
- Map to connectivity graph
- OpenPulse
- Algorithm
- H Gates
- Cancel Redundant Gates
- Translate to Qobj
- Measure then Repeat
- Supported
- JQ Gates
- Simulator
- Qobj
Challenges for near-term quantum computing

1. Don’t have many qubits

2. Can’t do many gates
   - **Gate error:** gates are imperfect
   - **Relaxation:** qubits do not retain state for long
Bernstein-Vazirani Algorithm*

Input (query)

\[ x_{n-1} \cdots x_1 x_0 \]

Secret Bitstring

\[ s_{n-1} \cdots s_1 s_0 \]

Output (result)

\[ x_{n-1} s_{n-1} \oplus \cdots \oplus x_1 s_1 \oplus x_0 s_0 \]

Optimal classical strategy:

\[
\begin{align*}
X &= 1 \ 0 \ \cdots \ 0 \ 0 \ (2^{n-1}) \\
X &= 0 \ 1 \ \cdots \ 0 \ 0 \ (2^{n-2}) \\
&\vdots \\
X &= 0 \ 0 \ \cdots \ 1 \ 0 \ (2) \\
X &= 0 \ 0 \ \cdots \ 0 \ 1 \ (1)
\end{align*}
\]

n tries

*E. Bernstein & U. Vazirani, STOC, 93
Bernstein-Vazirani Solution

Wherever there’s CNOT (i.e. the secret bitstring has a 1), phase kickback puts that control qubit in state |1>. 
Effect of Gate Errors and the Role of Software

Programmed Circuit

Good Compiled Circuit #1

Bad Compiled Circuit #2

Device

Win $5000 in prizes
Deadline: 15th May 2023
QISKit
Live Demo

Available here:
https://github.com/ajavadia/qiskit-sdk-py/blob/Demo/demo/Relaxation%20Demo.ipynb
Find qubit relaxation rate by running circuits

1. Put qubit in excited state and wait *variable amounts of time*, then measure.

2. Repeat each circuit many times (e.g. 1000 shots) to approximate probability of unwanted |0> state in each.

3. Find relaxation rate by fitting an exponential decay curve to the data.
Long-Term Road to Fault-Tolerant QC

Need automated tools to optimize in this design space & choose best design.

Factors:
- **Factoring**
- **Quantum Chemistry**
- **Quantum Machine Learning**

**Desired # Qubits:** $10^2 \ldots 10^8$
**Desired Logical Error** $P_L \approx 10^{-15} \ldots 10^{-5}$

**Available # Qubits:** $10 \ldots 100$
**Available Physical Error** $P_P \approx 10^{-5} \ldots 10^{-2}$

**Plaquette-Based Surface Code**
**Hole-Based Surface Code**

**Superconductors**

**Quantum Dots**

**Hard to scale, reduce noise.**

**Quantum Application (logical)**

**Error Correction**

**Quantum Technology (physical)**

**Only beats classical algs at scale.**

**Large gap b/w available and needed resources:**
**Must aggressively optimize.**

**Large gap b/w error rates: error corr. dominates**
**Must choose least expensive error correction.**
Best Design Has Minimum Resource Usage

Main questions:
1) How many qubits and time resources does a quantum circuit require for a particular design?
2) To what extent can a quantum circuit be optimized?
3) Which of the best designs should be used for many?

Scaffolds:
- Quantum Applications & Libraries
- Compilation & Logical Resource Estimation
- Intermediary Format
- Mapping Optimization: Reduce Total Moves
- Error Correction
- Network Optimization: Reduce Congestion

Resources:
github.com/epiqc/ScaffCC/
Space Overhead of Error Correction

**Plaquette-Based**
- Uses fewer physical qubits.

**Hole-Based**
- Uses more physical qubits.
Time Overhead of Error Correction

**Plaquette-Based: 2-step communication**
- Qubits slowly move close to each other to interact.
- EPRs are decoupled from data: can be prefetched.

**Hole-Based: 1-step communication**
- Braids move fast: \( n \) hops per cycle.
- Braids can’t be pre-fetched.
Application Dictates Code Favorability

<table>
<thead>
<tr>
<th>Error Correction Type</th>
<th>Communication Method</th>
<th>Space (Qubits)</th>
<th>Time (Latency)</th>
<th>Pre-fetchable?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plaquette-Based</td>
<td>Teleportations</td>
<td>Low</td>
<td>High</td>
<td>Yes</td>
</tr>
<tr>
<td>Hole-Based</td>
<td>Braids</td>
<td>High</td>
<td>Low</td>
<td>No</td>
</tr>
</tbody>
</table>

Cross-over point:
The computation size at which hole-based has better (lower) space × time, compared to plaquette-based.

Cross-over point occurs much later for parallel apps:
Plaquette-based code stays better for longer. Due to ability to schedule EPR pre-distribution around congestion.

Tools are needed for these insights:
Much of prior work had assumed hole-based to be better by default.
Co-Design of Applications, QEC Codes, Devices

Co-design for Maximum Benefit:

+ Very different technologies, with different constraints
+ Very different applications, with different characteristics
+ Very different Error Correction Codes, with different overheads

![Graph](image-url)

*Javadi-Abhari et al. MICRO 17*
Get Involved!

qiskit.org

Learn more

github.com/qiskit/qiskit-tutorial

Contribute code

github.com/qiskit

Help define

qiskit.slack.com

IBM Q Awards  Developer Challenge

Win $5000 in prizes
Deadline: 15th May 2018