



iQuHack 2024

Qubit Noise Reduction **Challenge**

Team Case-qit



Presentation Overview



01

Approach

How did we attack this problem?

02

Implementation

How did we carry out those strategies?

03

Testing

Did our scheme outperform the default?

04

Conclusion

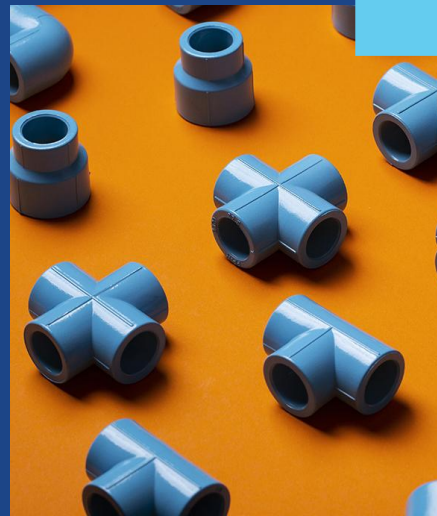
Where do we go from here?





The Challenge

1. Create a noise aware compilation scheme using AWS Braket
2. Remap quantum circuits to make the best use of high fidelity qubits
3. Improve on the performance of an existing quantum algorithm



01

Our team's Approach

Receive Quantum
Circuit as Input



Gather Counts of
Each Qubit's Usage

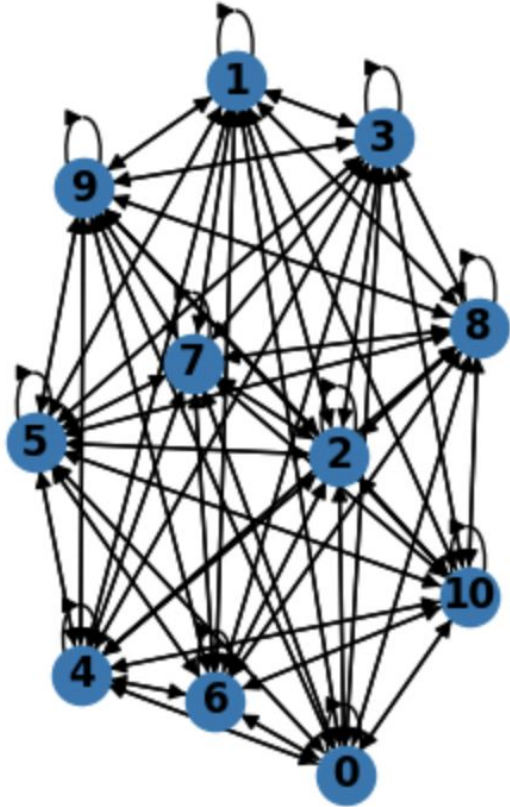


Map Hardware Qubits
According to Fidelity Values



Reassign Accurate Qubits to
High Usage Tasks





Using Topography To Graph Qubits

Nodes in our graph represent a qubit with edges connecting pairs of qubits. Our graph is weighted with each weight value being determined by our formula above.





Our initialization of a multiple digraph allowing loops.

```
import networkx as nx
graph = nx.MultiDiGraph()
graph.add_weighted_edges_from(edge_list)
for i in graph.nodes():
    graph.add_edge(i,i,0)
```



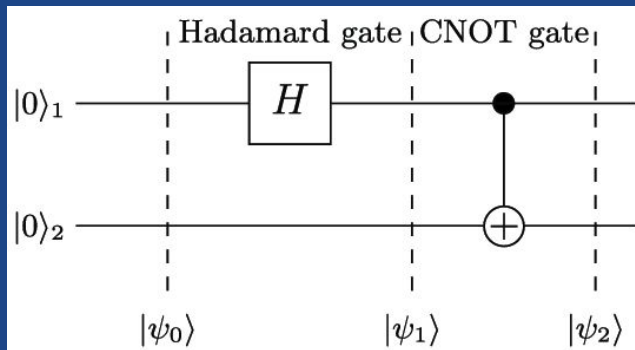
Our Graph Theory Approach

We used Networkx to create a graph with qubits as nodes and their connections + fidelities as weighted edges and then optimized via the degree of nodes.



Identifying Problem Qubits

Our algorithm therefore identifies qubit 0 as the more error prone qubit



Will then remap the highest fidelity qubit for it after map identification

Qubit 0 Counts:

2

Including both the hadamard gate and the cNOT gate

$$|\beta_{01}\rangle = \frac{|01\rangle + |10\rangle}{\sqrt{2}} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}; \quad |\beta_{10}\rangle = \frac{|00\rangle - |11\rangle}{\sqrt{2}} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ -1 \end{pmatrix};$$

Qubit 1 Counts:

1

Including only the cNOT gate



Simulation

Running the Braket_dm Simulator Tool

```
device = LocalSimulator('braket_dm')  
result = device.run(c, shots=1000).result()
```

Optimization

Remapping Qubits with those with Higher Fidelity

```
q[0] = 2  
q[1] = 3
```


```
nm1 = noise_model()  
newC = nm1.apply(newC)
```



Error Prevention

Avoiding Recompilation with Verbatim Comp

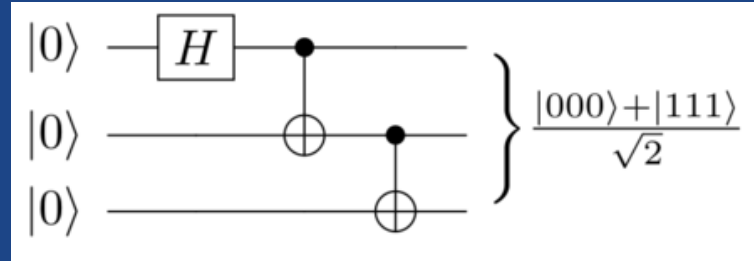
```
newC = Circuit().add_verbatim_box(Circuit().h(q[0]).cnot(q[0], q[1]))
```



03

Greenberger-Horne-Zeilinger State (GHZ)

Testing the
project





04

Our Results

Noise model before and
after qubit remapping





Before:


```
q0 : -H-DEPO(0.0019)-C-DEPO(0.074)-----
      | |
q1 : -----X-DEPO(0.074)-C-DEPO(0.072)-
      | |
q2 : -----X-DEPO(0.072)-
```

After:

```
q0 : -StartVerbatim-----X-DEPO(0.068)-EndVerbatim-
      |
q1 : -|-----H-DEPO(0.0019)-C-DEPO(0.068)-|-----|
      | | |
q2 : -*****-----X-DEPO(0.068)-C-DEPO(0.068)-*****-
```

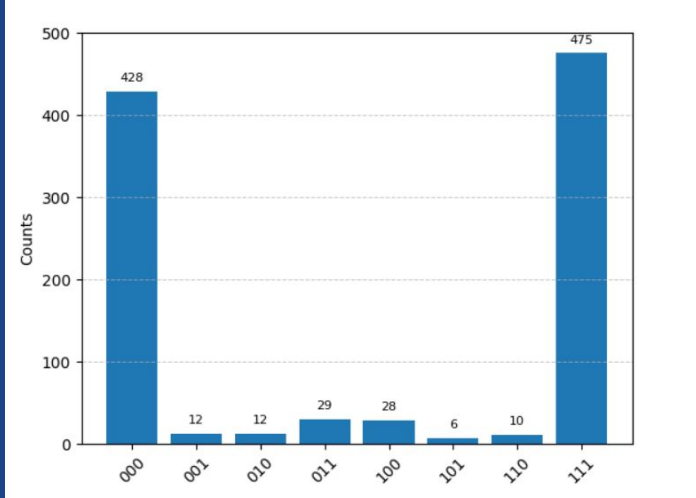
H-gate: 0.0019 → 0.0019 Cnot: 0.0074 → 0.0068

The simulated noise values decrease after our qubit remapping, demonstrating the more efficient allocation of high fidelity qubits



Testing Data

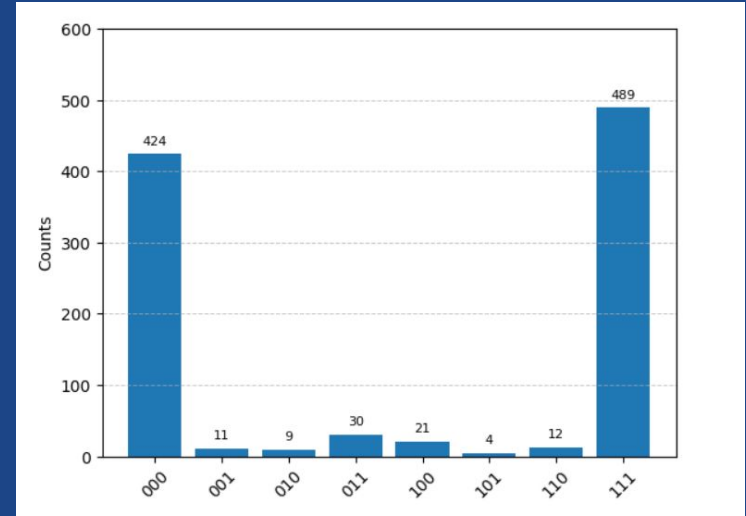
Data on Counts:



Unmapped Qubits

11%
Reduction In
Erroneous
output

SWAP



Mapped Qubits

Path Forward

We were obviously limited by time and hardware for this project so we would like to acknowledge some possible limitations that we have addressed.

- Fully connected qubit matrix
- Wanted to include epsilon into our calculations
- No actual hardware test
- Edge weight calculation variables are limited in our limitation
- All gate errors evaluated as equal



Thank you!

