WELCOME TO TUTORIAL

Session 4.1 Janus-FEM: Fast and Accurate Quantum Readout Calibration Using the Finite Element Method

ACES Lab

ADVANCED COMPUTING AND EMERGING SERVICE LAB

JanusQ Cloud

浙江大学 计算机科学与技术学院
COLLEGE OF COMPUTER SCIENCE AND TECHNOLOGY
ZHEJIANG UNIVERSITY

https://janusq.github.io/tutorials/

College of Computer Science and
Technology,
Zhejiang University

ASPLOS 2024

Outline of Presentation

Quantum readout is an operation to **read the information from quantum bits to classical bits**.

Measured bitstrings in different samples

Probability distribution

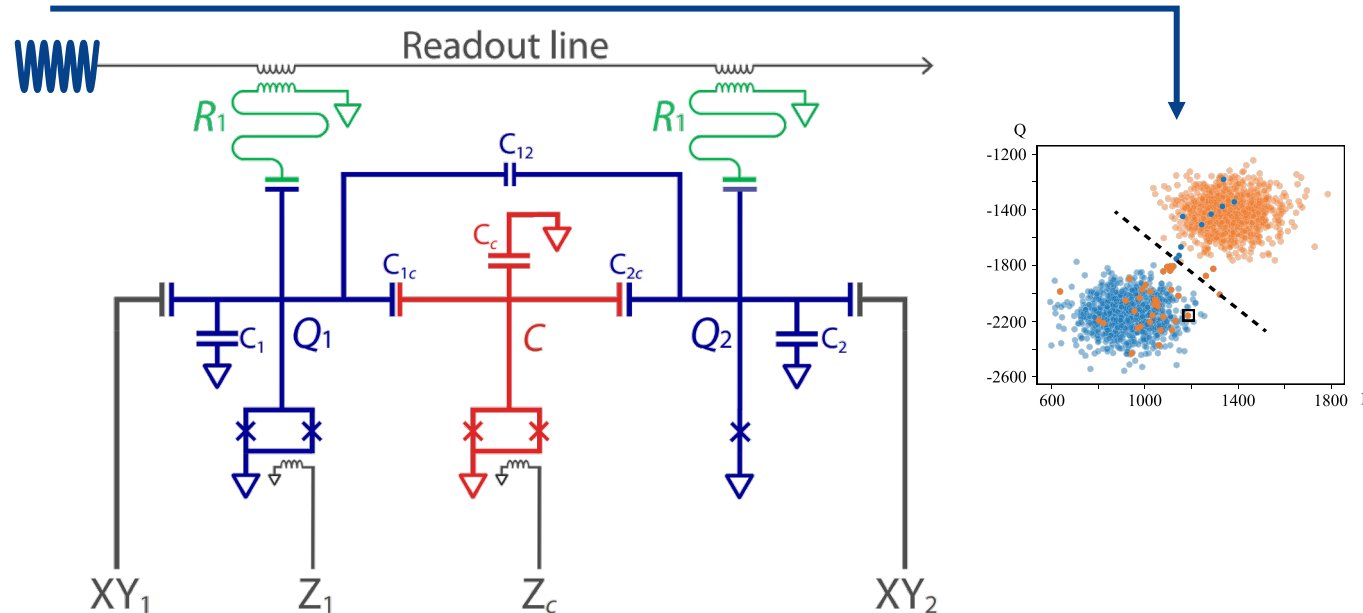**Readout error** is significant on current quantum hardware.
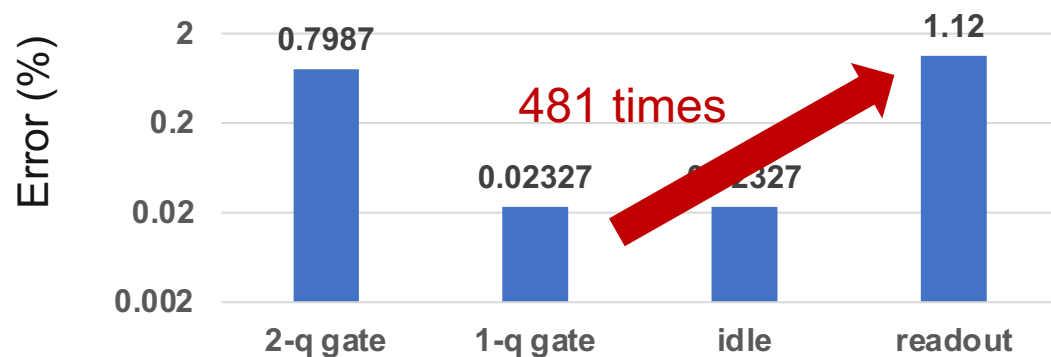
Noise on 127-qubit IBM Sherbrooke quantum device
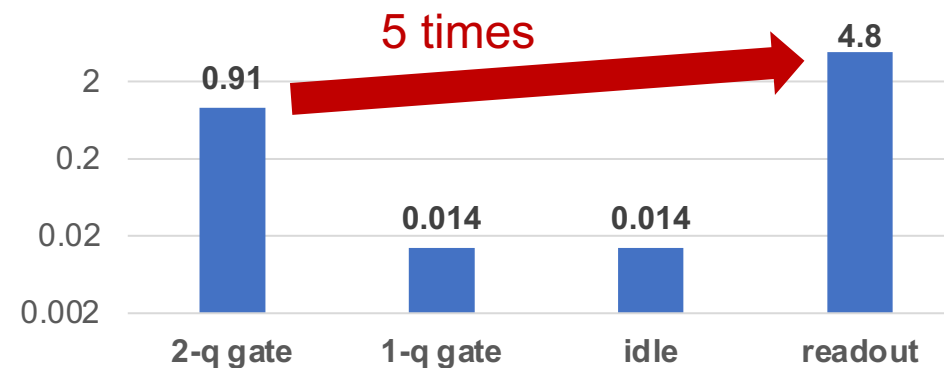
Noise on 10-qubit Tianmu quantum device

Implementation of readout on superconducting qubits

Source of readout error

(1) Pulse in   *(2) Pulse takes the information out.*   *FFT*

Readout line

R_1 R_1

C_{12}

C_c

C_{1c} C_{2c}

C_1 Q_1 C Q_2 C_2

XY_1 Z_1 Z_c XY_2

Q
-1200
-1400
-1800
-2200
-2600
600   1000   1400   1800   I

from 0 to 1

Q1 ➡ **Q1**

Excitation error

from 1 to 0

Q1 ➡ **Q1**

Relation error

Q2

from 1/0 to 0/1

Q1 ➡ **Q1**

Crosstalk
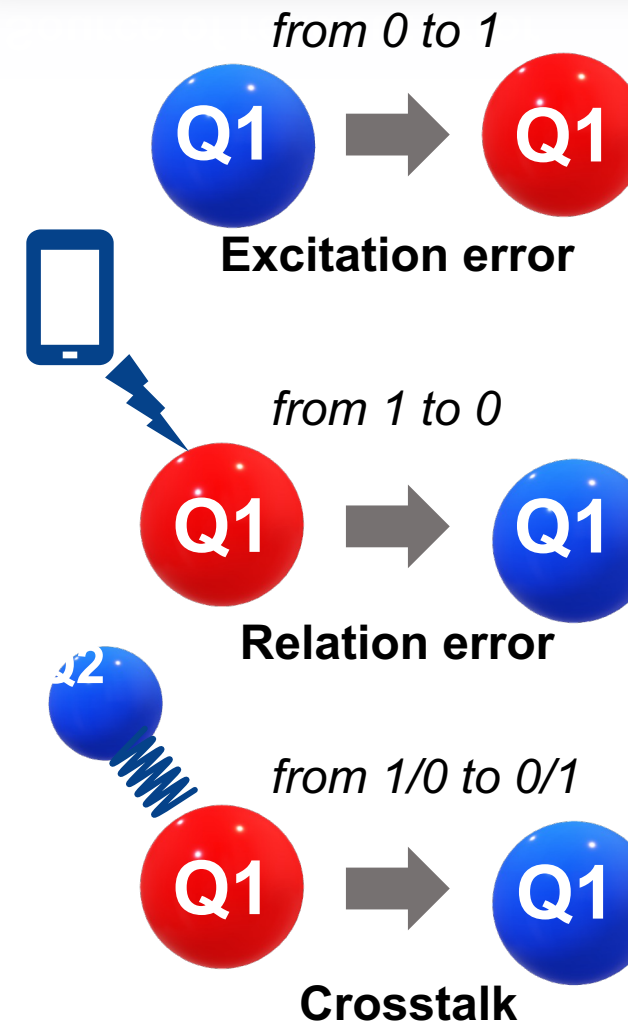
State-dependent Readout Error

Readout errors vary in different combination of measured qubits

Crosstalk has different frequencies when Q2 is measured 0, 1 or not measured

Similar to entanglement
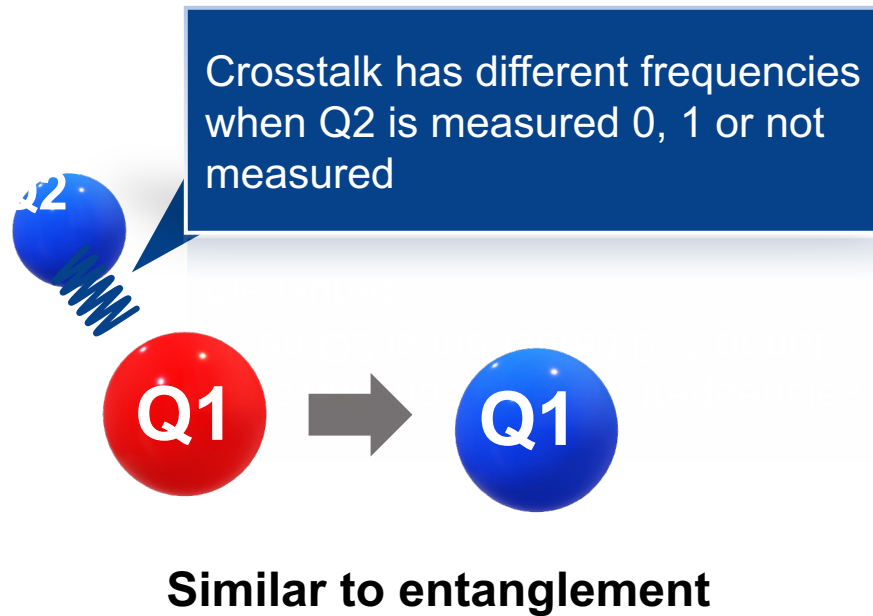
Example of state-dependent and readoutdependent noises on the IBMQ Perth quantum device.

the error of q_1 depends on its own operation.

the error also depends on the readout of q_5.

error of q_1 (%)

1.44
1.44
1.36
1.32

0 1 ∅
state of q_1

0 1 ∅
state of q_2

0 1 ∅
state of q_3

0 1 ∅
state of q_4

0 1 ∅
state of q_5

Q2

Q1 ➡ Q1

5

ASPLOS 2024

Background
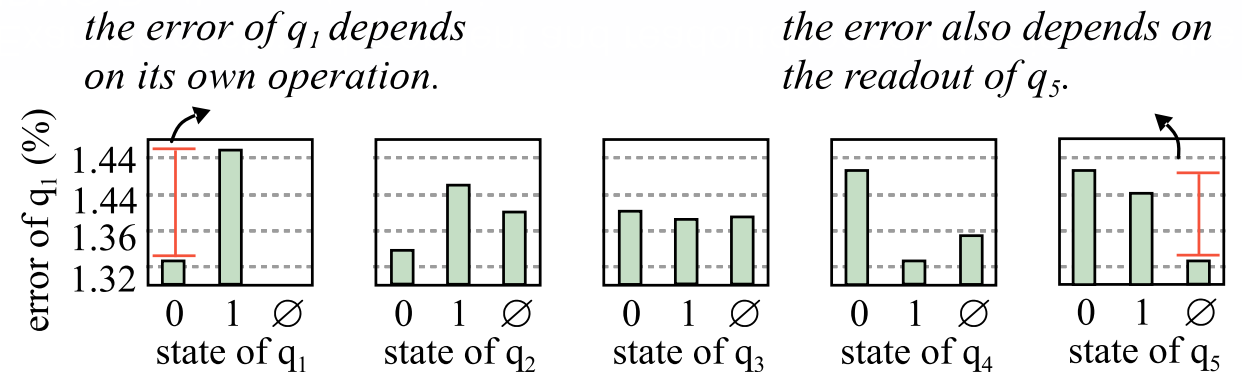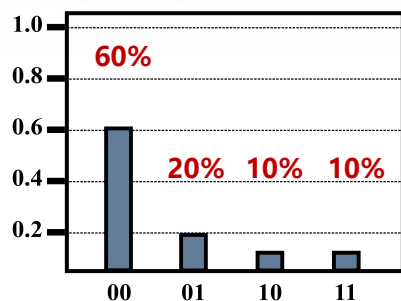
Ideal readout

State vector → **Ideal distribution (ideal program output)**

$$\begin{bmatrix} 0.6 \\ 0.2 \\ 0.1 \\ 0.1 \end{bmatrix}$$

Readout with noise
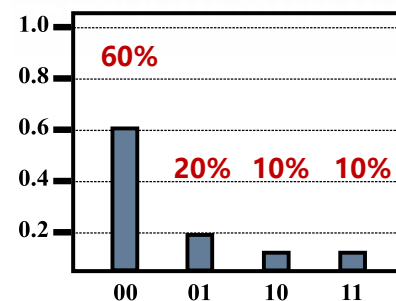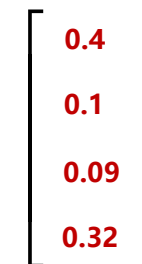
State vector → **Noisy distribution (noisy program output)**

$$\begin{bmatrix} 0.4 \\ 0.1 \\ 0.09 \\ 0.32 \end{bmatrix}$$

Matrix-based readout error calibration
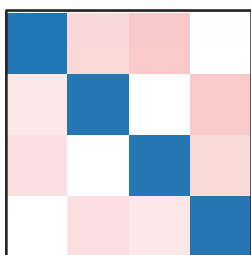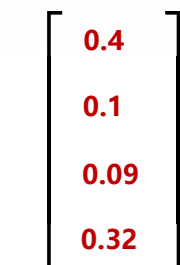
Calibration matrix · $\begin{bmatrix} 0.4 \\ 0.1 \\ 0.09 \\ 0.32 \end{bmatrix}$ **Noisy distribution** $=$ $\begin{bmatrix} 0.6 \\ 0.2 \\ 0.1 \\ 0.1 \end{bmatrix}$ **Calibrated distribution**
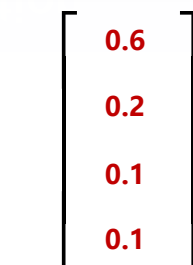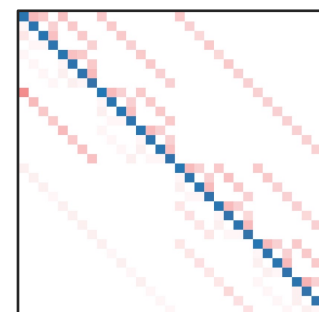

Calibration matrix of a 5-qubit readout $2^5 \times 2^5$

The size exponentially increases!

Basic Matrix-based readout calibration

Step 1. Matrix characterization

Prepares qubits to different basis states and apply measurement.

q_1 — prepare to 00 — (measure)
q_2 —

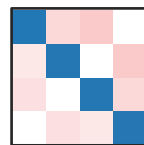q_1 — prepare to 01 — (measure)
q_2 —

Fill in a noise matrix.

$$M = \begin{bmatrix} 0.6 & 0.1 & 0.2 & 0.1 \\ 0 & 0.7 & 0.2 & 0.1 \\ 0.2 & 0.2 & 0.6 & 0 \\ 0 & 0.1 & 0.1 & 0.8 \end{bmatrix}$$
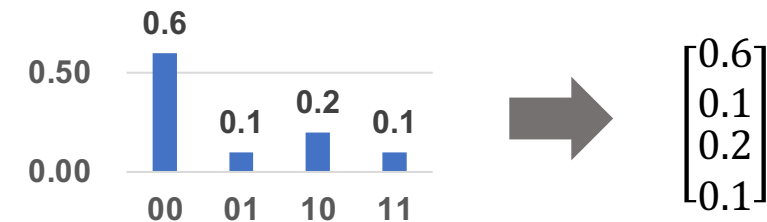
Inverse the noise matrix

$$M^{-1} =$$

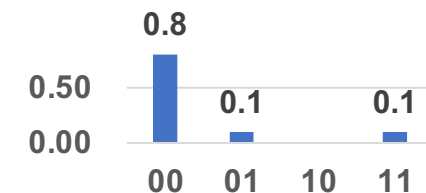

calibration matrix

Step 2. Calibration for any input

Represent the measured distribution as a vector.

$$\begin{bmatrix} 0.6 \\ 0.1 \\ 0.2 \\ 0.1 \end{bmatrix}$$

Apply matrix-vector multiplication.

$$\begin{bmatrix} 0.6 & 0.1 & 0.2 & 0.1 \\ 0 & 0.7 & 0.2 & 0.1 \\ 0.2 & 0.2 & 0.6 & 0 \\ 0 & 0.1 & 0.1 & 0.8 \end{bmatrix}^{-1} \cdot \begin{bmatrix} 0.6 \\ 0.1 \\ 0.2 \\ 0.1 \end{bmatrix} = \begin{bmatrix} 0.8 \\ 0.1 \\ 0 \\ 0.1 \end{bmatrix}$$

$$M^{-1} \cdot P_{\text{read}} = P_{\text{calibrated}}$$

Basic Matrix-based readout calibration

Step 1. Matrix characterization

Prepares qubits to different basis states and apply measurement.

2^N circuits are executed to measure qubits on all basis states.

Fill in a noise matrix.

The size of the noise matrix is $2^N \times 2^N$.

Inverse the noise matrix

Calcaute the inverse has $\mathcal{O}(4^N)$ complexity.

Step 2. Calibration for any input

Represent the measured distribution as a vector.

Transformation has linear complexity.

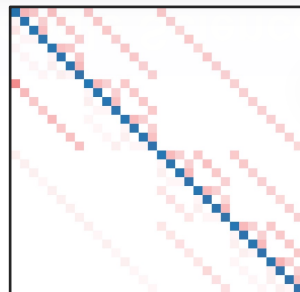**Apply matrix-vector multiplication.**

Multiplication has $\mathcal{O}(4^N)$ complexity.

8.8 TB and 10 hours for a 32-qubit calibration on a server with xxxx

Limitations of Current Methods

IBU (Google Science 2021) Realizing topologically ordered states on a quantum processor.
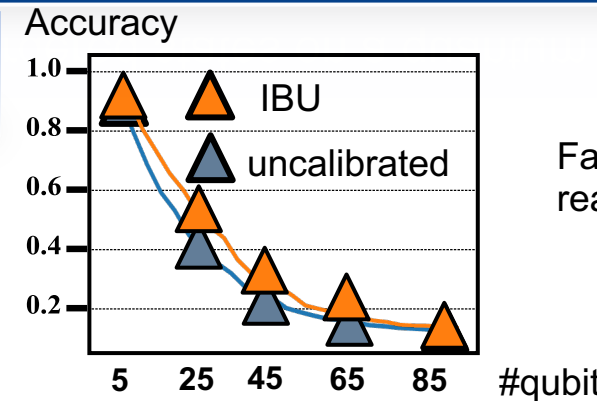

Real calibration matrix   Single-qubit matrix

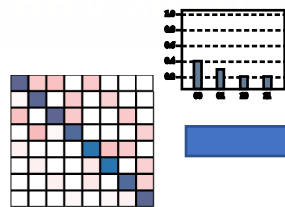*Crosstalk makes the matrix not simple tensor-product result.*

use tensor product of a series of single-qubit meta-matrices

Accuracy
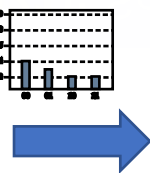
IBU

uncalibrated

#qubit

Fail to calibrate on 80-qubit readout output

Fast but not accurate: ignore the qubit interactions.
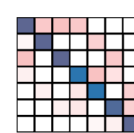
M3 (IBM PRA 2021): Scalable mitigation of measurement errors on quantum computers

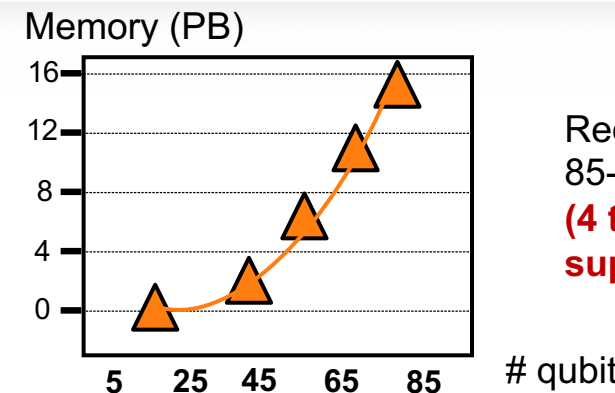Before pruning   Pruning based on program output   After pruning

use a sparsity-aware method prune the matrix under a threshold of Hamming distance

Memory (PB)

qubit

Require 16PB to calibrate a 85-qubit result.
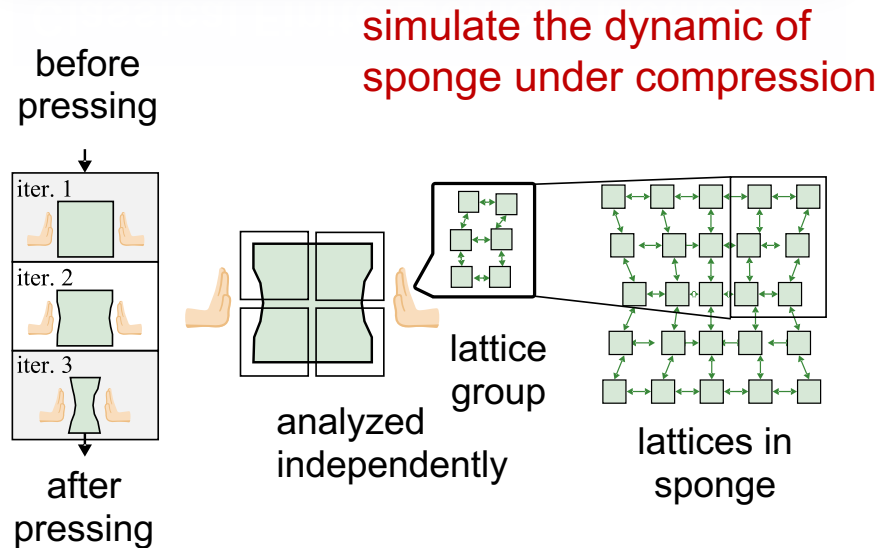**(4 times the Fugaku supercomputer)**

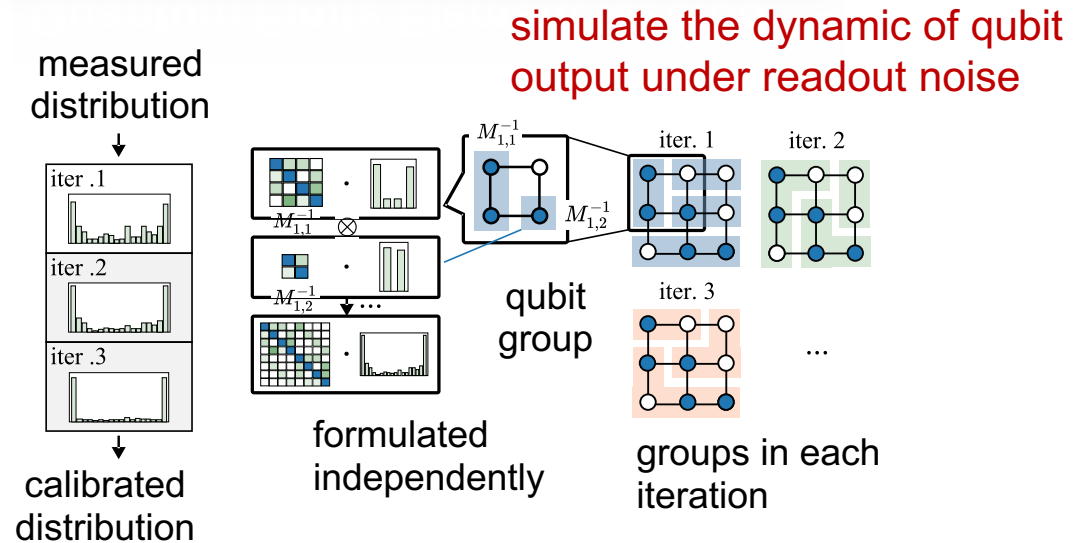**Accurate but not fast:** many matrix elements cannot be ignored

Outline of Presentation

- Background ad challenges

- **Overview of Janus-FEM**

- Janus-FEM Characterization And Calibration

- Experiment

- API Of Janus-FEM

Calibration based on Finite Element method (FEM)

Classical Finite Element Method

simulate the dynamic of sponge under compression

before pressing

iter. 1

iter. 2

iter. 3

after pressing

analyzed independently

lattice group

lattices in sponge

Quantum Finite Element Method

simulate the dynamic of qubit output under readout noise

measured distribution

iter .1

iter .2

iter .3

calibrated distribution

$M_{1,1}^{-1}$

$M_{1,1}^{-1}$

$M_{1,2}^{-1}$

$M_{1,2}^{-1}$

qubit group

iter. 1

iter. 2

iter. 3

...

formulated independently

groups in each iteration

① partitions the sponge into lattices

② analyzes the state of each lattice independently

③ simulate the interaction

④ update the state of sponge
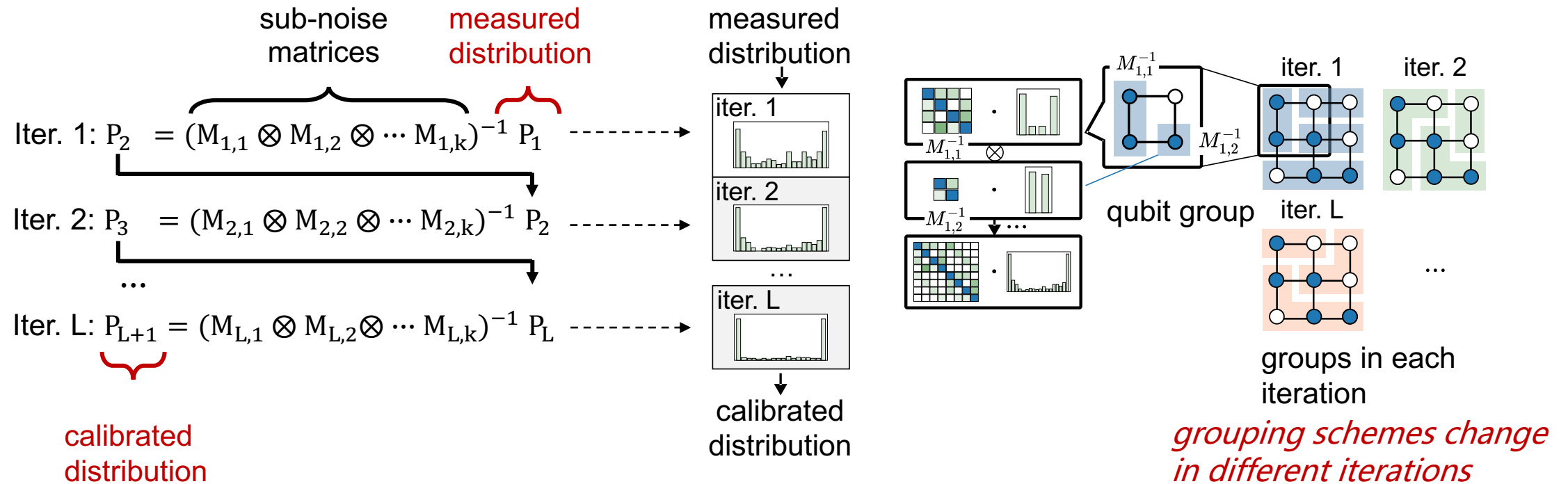
① partitions qubits into groups

② analyze the noise in each group independently

③ simulate the interaction

④ update the calibration result of qubits

A **divide-and-conquer strategy** to calibrate measured distribution

QuFEM reformulates the calibration as an iterative process with a series of sub-noise matrices.

sub-noise matrices · measured distribution · measured distribution

Iter. 1: $P_2 = (M_{1,1} \otimes M_{1,2} \otimes \cdots M_{1,k})^{-1} P_1$

Iter. 2: $P_3 = (M_{2,1} \otimes M_{2,2} \otimes \cdots M_{2,k})^{-1} P_2$

...

Iter. L: $P_{L+1} = (M_{L,1} \otimes M_{L,2} \otimes \cdots M_{L,k})^{-1} P_L$

calibrated distribution

iter. 1

iter. 2

...

iter. L

calibrated distribution

$M_{1,1}^{-1}$ · $M_{1,2}^{-1}$ ⊗ · ...

qubit group

iter. 1 iter. 2 iter. L

...

groups in each iteration
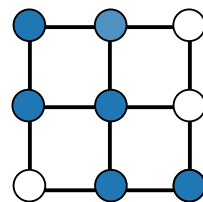
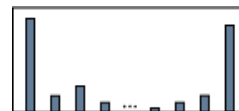*grouping schemes change in different iterations*

An example

Input :

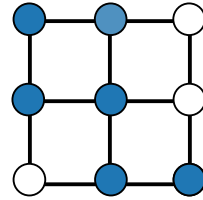measured qubits
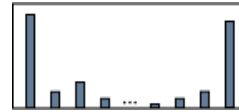
measured distribution
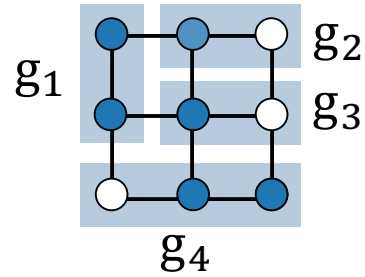
An example

Input :

measured qubits
measured distribution

Iteration 1 :

grouping scheme
computation formulation

$$P_2 = (M_{1,1} \otimes M_{1,2} \otimes M_{1,3} \otimes M_{1,3})^{-1} \cdot P_1$$
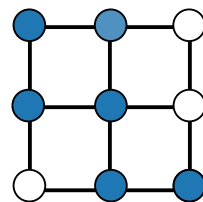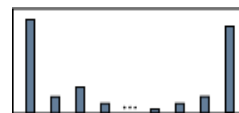
Matrices are generated according to the measured qubits.

Input :

measured qubits

measured distribution

Iteration 1 :

grouping scheme

g_1 g_2 g_3 g_4
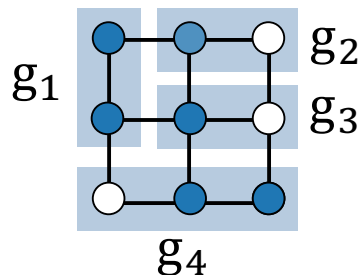
computation formulation

$$P_2 = (M_{1,1} \otimes M_{1,2} \otimes M_{1,3} \otimes M_{1,3})^{-1} \cdot P_1$$

Matrices are generated according to the measured qubits.

Iteration 2 :

grouping scheme

g_1 g_2 g_3 g_4
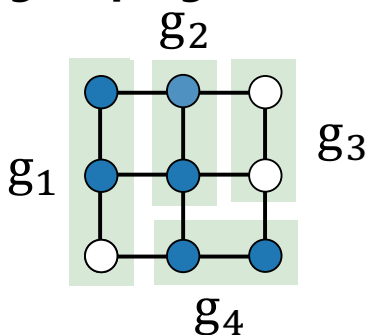
computation formulation

$$P_3 = (M_{2,1} \otimes M_{2,3} \otimes M_{2,4})^{-1} \cdot P_2$$

Outline of Presentation

- Background and challenges

- Overview of Janus-FEM

- **Janus-FEM Characterization And Calibration**

- Experiment

- API Of Janus-FEM

Deployment on Specific Quantum Device

Technique 1: determine the grouping scheme

Data collection

Run benchmarking circuits.

Possible states of a qubit in a benchmarking circuit:

- 1: qubit is set 0 and measured

- 2: qubit is set 1 and measured

- 3: qubit is set 0 or 1 and not measured

Not all qubits are measured to maximize the variety.

Deployment on Specific Quantum Device

Technique 1: determine the grouping scheme

Data collection

Run benchmarking circuits.

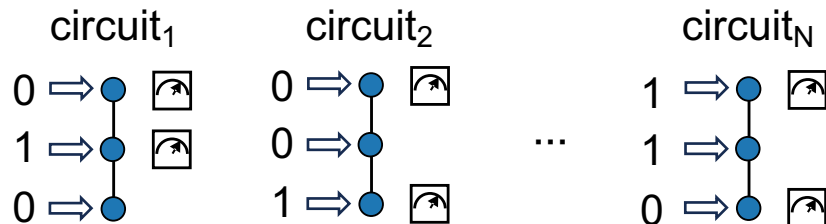Possible states of a qubit in a benchmarking circuit:

- 1: qubit is set 0 and measured
- 2: qubit is set 1 and measured
- 3: qubit is set 0 or 1 and not measured

Not all qubits are measured to maximize the variety.

Qubit partition

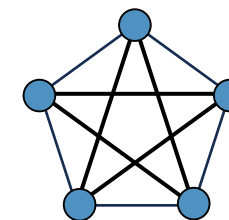Characterize the **interaction** from one qubit to another qubit under different states:

$$\text{interact}(q_i.\text{state} = x \rightarrow q_j.\text{state} = x)$$

$$= \left| P(q_j.\text{error} = 1 | C1, C2) - P(q_j.\text{error} = 1 | C2) \right|$$

error rate of q_j under $C1, C2$ average error rate of qz_j

$$C1: q_i.\text{state} = x, \qquad C2: q_j.\text{state} = y$$
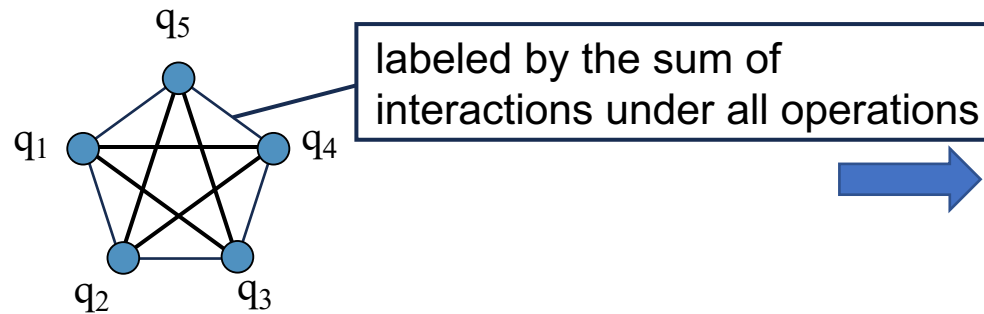
Construct weighted graph

Deployment on Specific Quantum Device

Technique 1: determine the grouping scheme

Qubit partition

Construct a **weighted qubit graph**:

q_5

q_1   q_4

> labeled by the sum of interactions under all operations

q_2   q_3

Partitions with a **MAX-CUT solver**:
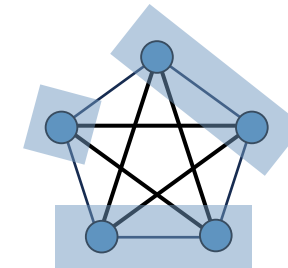
try to comprehensively capture the interactions between qubits

An Example   **Prior knowledge of hardware helps grouping**

Readout resonator 1

① — ② — ③
⑧ — ⑦ — ⑥ — ⑤ — ④

Readout resonator 2

⑨ — ⑩ — ⑪ — ⑫ — ⑬
⑱ — ⑰ — ⑯ — ⑮ — ⑭
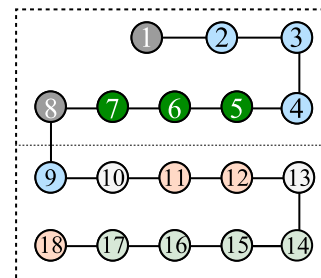
18-qubit topology

group 1:   ⑭   ⑮   ⑯   ⑰
same readout resonator

group 2:   ②   ③   ④   ⑨
similar readout frequency

group 3:   ①   ⑧
overlapping frequency shift region

Deployment on Specific Quantum Device

Technique 2: sub-noise matrix generation

Perform matrix-vector multiplication

$$\text{Iter. i: } P_{i+1} = (M_{i,1} \otimes M_{i,2} \otimes \cdots M_{i,k})^{-1} P_i$$

Matrix generation

Noise matrix formulates the transformation probability from the ideal state to measured state.

$$\begin{bmatrix} 0.6 & 0 & 0.1 & 0 \\ 0.1 & 0.7 & 0.2 & 0.1 \\ 0.2 & 0.2 & 0.6 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.8 \end{bmatrix} =$$

set state

read state	00	01	10	11
00	0.6	0	0.1	0
01	0.1	0.7	0.2	0.1
01	0.2	0.2	0.6	0.1
11	0.1	0.1	0.1	0.8

Deployment on Specific Quantum Device

Technique 2: sub-noise matrix generation
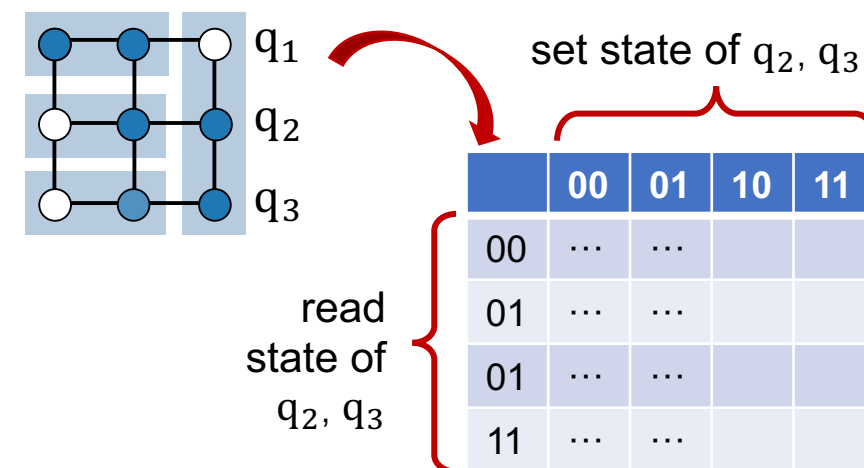
Perform matrix-vector multiplication

$$\text{Iter. i: } P_{i+1} = (M_{i,1} \otimes M_{i,2} \otimes \cdots M_{i,k})^{-1} P_i$$

Matrix generation

Noise matrix formulates the transformation probability from the ideal state to measured state.

Sub-noise matrices of QuFEM formulates the transformation probability of states inside the qubit groups.

set state

	00	01	10	11
00	0.6	0	0.1	0
01	0.1	0.7	0.2	0.1
01	0.2	0.2	0.6	0.1
11	0.1	0.1	0.1	0.8

$$\begin{bmatrix} 0.6 & 0 & 0.1 & 0 \\ 0.1 & 0.7 & 0.2 & 0.1 \\ 0.2 & 0.2 & 0.6 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.8 \end{bmatrix} = \text{read state}$$

set state of q_2, q_3

	00	01	10	11
00	...	...		
01	...	...		
01	...	...		
11	...	...		

read state of q_2, q_3

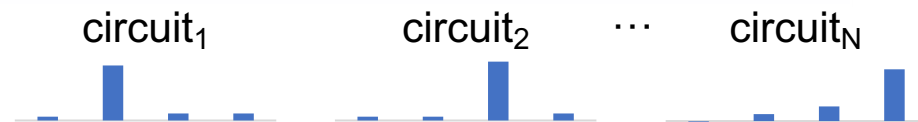$$M[x][y] = P(\{q_2, q_3\}. \text{read} = x | \{q_2, q_3\}. \text{set} = y, q_1 = 2)$$

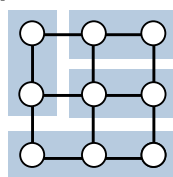**Transformation probability when q_1 is not measured**

Put all together

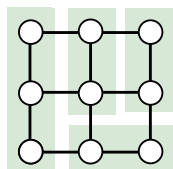## Characterization

Iteration 1. *Run benchmarking circuits.*

circuit$_1$ circuit$_2$ $\cdots$ circuit$_N$

Iteration 1. *Partition qubits.*

Iteration 1. *Calibrate.*

distributions, grouping
scheme of iteration 1.

Iteration 2. *Partition qubits.*

distributions, grouping
scheme of iteration 2.

Iteration 2. *Calibrate.*

Calibration

Input. *measured qubits   measured distribution*
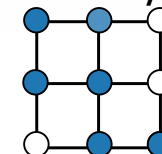
Iteration 1. *Generate sub-noise matrices.*

$M_{1,1}$ $M_{1,2}$ $M_{1,3}$ $M_{1,3}$
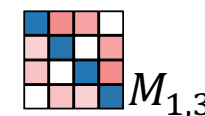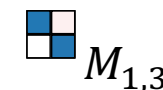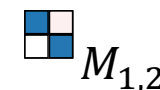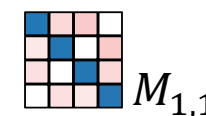
Iteration 1. *Calibrate.*

Iteration 2. *Generate sub-noise matrices.*

$M_{2,1}$ $M_{2,3}$ $M_{2,4}$
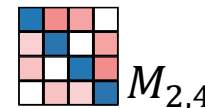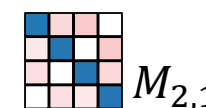
Iteration 2. *Calibrate.*

Output

Sparse Tensor-Product Engine

Observation

$$\left(\boxtimes \otimes \boxtimes \otimes \boxtimes \otimes \boxtimes \right) \cdot$$

A large number of sparse intermediate vectors is generated in the tensor-product.

Implementation

$M_{2,1}^{-1}$ $M_{2,2}^{-1}$

$$\left(\boxtimes \otimes \boxtimes \right) \cdot \begin{bmatrix} 0.47 \\ 0.00 \\ \cdots \\ 0.53 \\ 0.00 \end{bmatrix} P_1$$
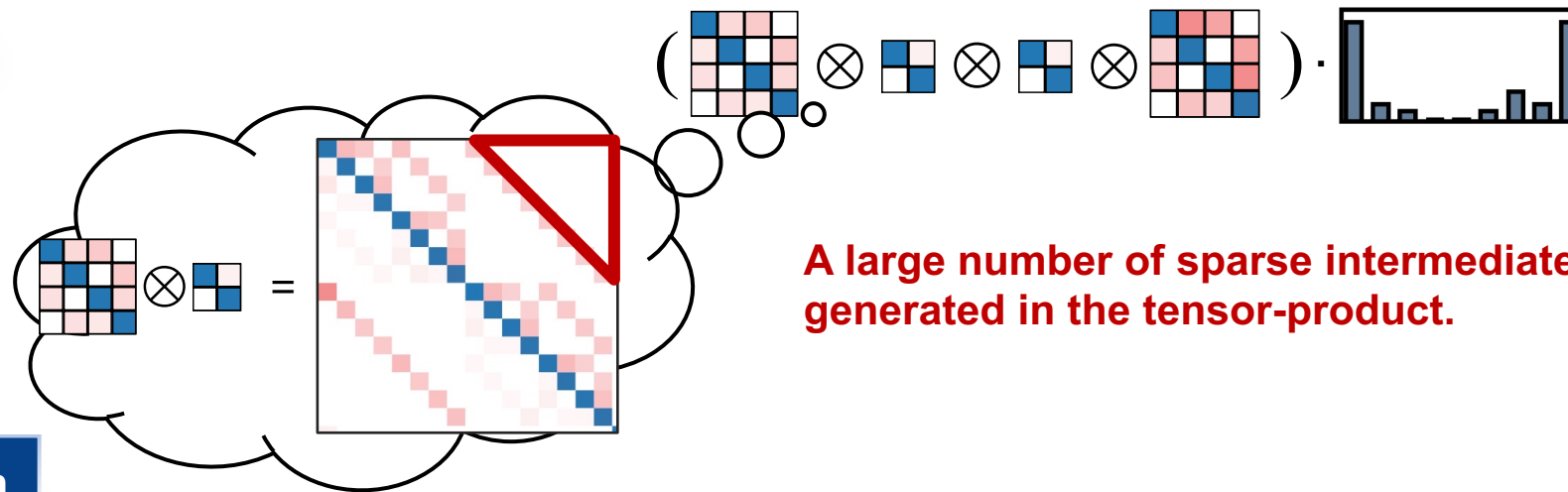
target equation

Use a key-value table to store sparse vector

x	$prob.$
$P_1(000)$	0.47
$P_1(011)$	0.53

① →

	$value$
00	0.50
01	-0.02
10	0.01
11	0.01

Calculate the tensor product

② $\otimes$

	$value$
0	0.99
1	0.01

$\cdot\ 0.47 =$

③ $|value| < \beta$

	$value$
000	0.49
001	0.01
010	-0.01
100	0.01
101	10⁻⁴
110	10⁻⁴

...

Aggregate the tensor-product result

④ → $\sum$ →

x	$prob.$
$P_2(000)$	0.48
$P_2(001)$	6×10^{-3}
$P_2(010)$	6×10^{-3}
$P_2(011)$	0.50
$P_2(111)$	6×10^{-3}

Prune values < threshold (e.g., 10^{-5})

Compute the tensor-product of other basis states

For each basis states
① calculate the matrix-vector multiplication
② calculate the tensor-product
③ prune intermediate values
④ sum intermediate values to obtain output.

Outline of Presentation

- Background and challenges

- Overview of Janus-FEM

  - Characterization

  - Calibration

- **Experiment**

- API Of Janus-FEM

Setup

Platform	#Qubits	1-q fidelity	2-q fidelity	Instructions
Quafu	136	94.6±3.1%	94.6±3.0%	ID,RX,RY,RZ,H,CX
	18	95.9±1.3%	95.9±1.3%	ID,RX,RY,RZ,H,CX
Rigetti	79	99.5±1.1%	90.0±6.4%	CPHASE,XY
Self-developed	36	99.9±0.1%	98.7±0.8%	U3,CZ
IBMQ	7	99.9±0.1%	99.2±0.1%	CX,ID,RZ,SX,X

Evaluated hardware

IBU: KJ Satzinger, et al. Realizing topologically ordered states on a quantum processor. Science 2021

CTMP: Sergey, et al. Mitigating measurement errors in multiqubit experiments. PRA 2021.

M3: Paul D Nation , et al. Scalable mitigation of measurement errors on quantum computers. PRX Quantum 2021.

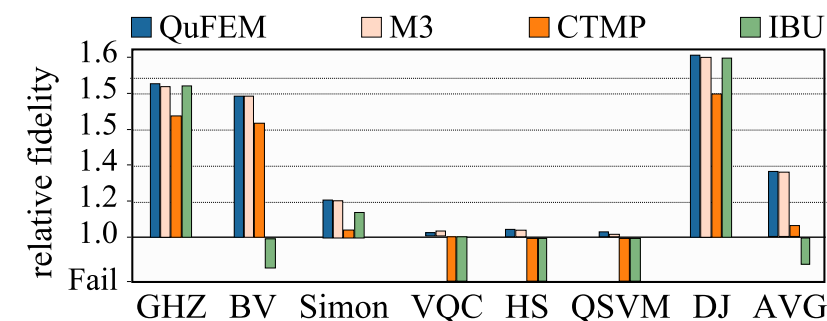**Q-BEEP:** Nathan Wiebe, et al. Qbeep: Quantum Bayesian error mitigation employing Poisson modeling over the hamming spectrum. ISCA 2023.

Baselines

Result

Achieving closed to $\mathcal{O}(N^2)$ **complexity**
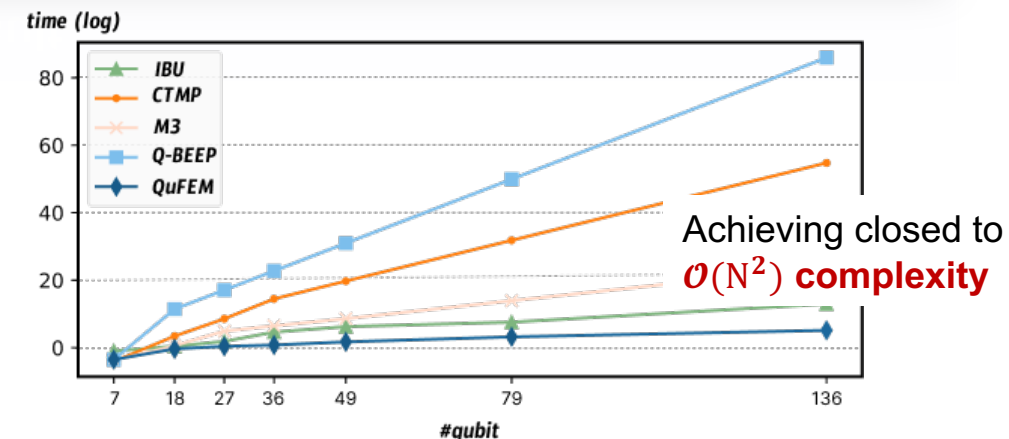
QuFEM reduces the calibration time of the 136-qubit program output from **119.44 hours** (IBU) to **169.65 seconds** (119.44 × **reduction**).

QuFEM shows an average improvement in relative fidelity of **1.003×, 1.2×,** and **1.4×** compared to M3, CTMP, and IBU, respectively.

Outline of Presentation

- Background and challenges

- Overview of Janus-FEM

    - Characterization

    - Calibration

- Experiment

- **API Of Janus-FEM**

API Of Janus-FEM

File:
- examples/3-2.readout_calibration_realqc.ipynb
- https://janusq.github.io/tutorials/Demonstrations/3-2.readout_calibration_realqc

Import package and data

```python
from janusq.optimizations.readout_mitigation.fem import Mitigator
from qiskit.quantum_info.analysis import hellinger_fidelity
from janusq.optimizations.readout_mitigation.fem.tools import npformat_to_statuscnt
from janusq.dataset import protocol_8 as benchmark_circuits_and_results, ghz_8qubit as ghz_output

qubits = 8
```

Construct mitigator

```python
mitigator = Mitigator(qubits, n_iters = 2)
scores = mitigator.init(benchmark_circuits_and_results, group_size = 2,multi_process=False, draw_grouping = True)
```

Calibrate GHz circuit output

```python
n_qubits = 4
outout_ideal = {'1'*n_qubits:0.5,'0'*n_qubits:0.5}
output_fem = mitigator.mitigate(ghz_output[0],[i for i in range(n_qubits)], cho = 1 )
output_fem = npformat_to_statuscnt(output_fem)
```

Compute the fidelity

```python
print("Janus-FEM fidelity: ",hellinger_fidelity(outout_ideal,output_fem))
```

Thanks for listening

Janus-FEM: Fast and Accurate Quantum Readout Calibration Using the Finite Element Method

Siwei Tan, Liqiang Lu*, Hanyu Zhang, Jia Yu, Congliang Lang, Yongheng Shang, Xinkui Zhao, Mingshuai Chen, Yun Liang, and Jianwei Yin*