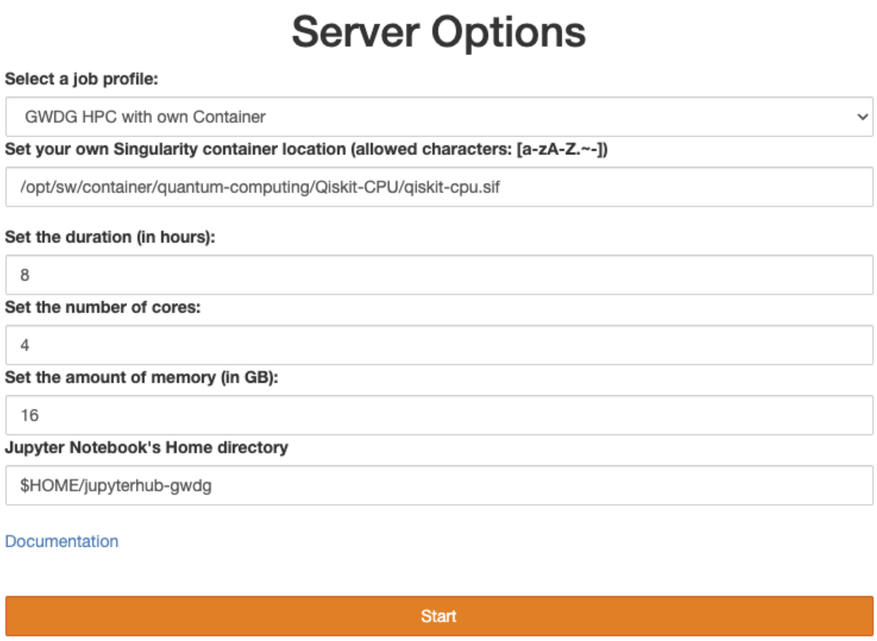
GWDG
AG-C
Lourens van Niekerk

Exercise 1 / April 4, 2024
Practical Course High-Performance Computing / SoSe 2024
60 Minutes Total

Contents

Task 1: Setup (5 min)

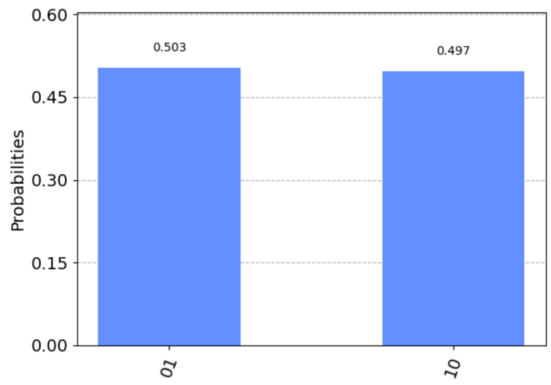
1. Login at `https://jupyter-hpc.gwdg.de/hub/login`.
2. At "Select a job profile", select `GWDG HPC with own container`.
3. Use `/opt/sw/container/quantum-computing/Qiskit-CPU/qiskit-cpu.sif` as the container path.
4. You don't need all that power and memory! Decrease the amount of cores and memory to 4 and 16, respectively. Check that input matches the figure below.



5. Spawn your server!
6. Upload the Jupyter notebook (`.ipynb`) from the `exercises` folder.
7. Examine and run `pchpc_qc_ex.ipynb`
8. (If you would prefer to set up a environment locally for qiskit, instructions are at the final task).

Task 2: Modified entanglement (10 min)

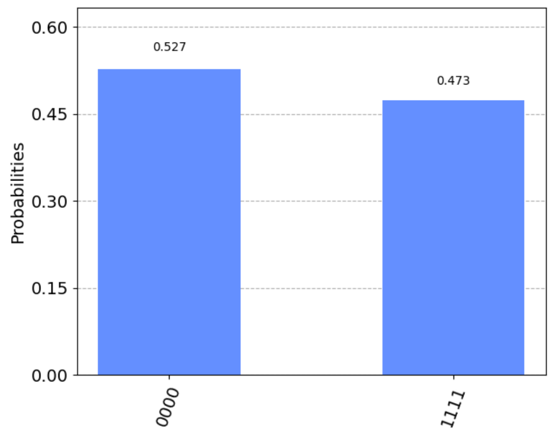
1. There exists an entangled state where 2 qubits are always in **different** states after measurement:



Implement the Quantum circuit producing that state

Task 3: 4-qubit entanglement (10 min)

1. Create a circuit containing 4 qubits and entangle all 4 such that if any of them are measured to be a 0 or 1, the rest collapse to be 0 or 1, respectively, as well.

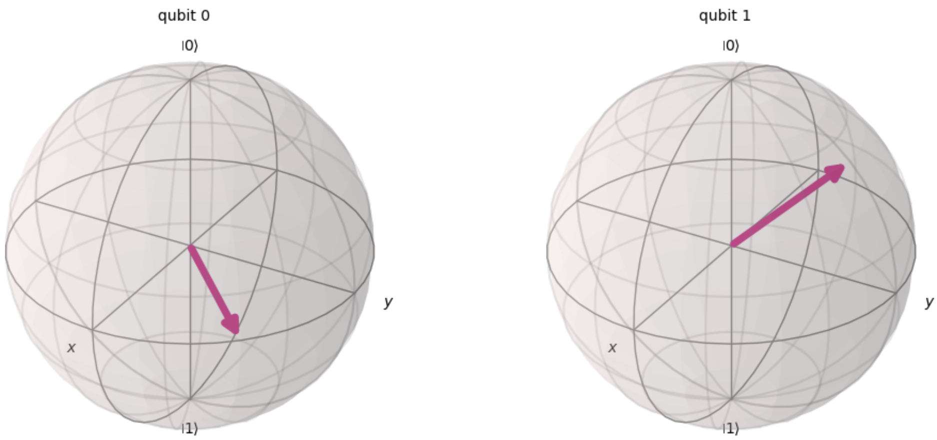


Hints

- Think iteratively about control gates

Task 4: Bloch sphere manipulation (10 min)

1. Create a 2-qubit circuit and use rotation gates to create the following Bloch sphere representation. To be extra sure, the statevector you should get is `Statevector([ 0.60355339-0.25j, 0.60355339+0.25j, 0.10355339+0.25j, -0.10355339+0.25j], dims=(2, 2))`



Hints

- The gates you want to use are rx, ry and rz.
- Their structures are `circuit.rgate(ROTATION, QUBIT)`
- Use `numpy.pi` for the degree of rotations.

Task 5: Phase encoding (5 min)

1. Use the phase function given and encode the number 127 into the phase of a single qubit.
2. Also encode the number 100 in the qubit. What is a good base for both numbers?
3. Compared to this single qubit, how many bits does it take encode these numbers?
4. How many qubits would it take to output a measure representing these numbers? Why?

Task 6: Deutsch-Josza algorithm (20 min)

1. Recall the algorithm:
   - Prepare input qubits each as $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ state
   - Prepare the output qubit as $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ state
   - Apply the oracle
   - Get input qubits into computational bases with Hadamard gates
   - Measure to test if function is constant or balanced.
2. Implement the Deutsch-Josza algorithm for 4 or more qubits (1 output qubit and 3 or more input qubits)

- Look at the circuit illustration in the slides for inspiration
- Use the given constant and balanced oracles
- Which oracle was used in your simulation?

Optional Task 7: Run your circuit on a real device (20 min)

This is a difficult **additional** task that will support your understanding in the topic.

1. The code for this is in the exercises
2. You need an account from IBM Quantum
3. Retrieve your token from the account and insert in (and uncomment) the according line in the notebook
4. After the first run, the token line can be commented

Optional Task 8: Set up local Qiskit environment (10 min)

This is a difficult **additional** task that will support your understanding in the topic.

1. Use pyenv or conda to create a new environment. Call it "qiskitenv".
2. We need the environment to be usable for jupyter notebooks, and create a kernel in which to put qiskit. In the case of conda, run the following commands:
   ```
   conda init bash
   conda .bashrc
   conda activate qiskitenv
   conda install notebook ipykernel
   python -m ipykernel install --user --name qiskitenv --display-name "QiskitEnv"
   pip install qiskit[visualization]
   ```
3. Run `jupyter notebook` on your terminal.
4. Now you can access the jupyter notebook exercises from the portal that appeared in your web browser.