Building your first CUDA Quantum Program

Python C++

We can define our quantum kernel as a typical Python function, with the additional use of the `@cudaq.kernel` decorator. Let's begin with a simple GHZ-state example, producing a state of maximal entanglement amongst an allocated set of qubits.

```
import cudaq

qubit_count = 2


# Define our kernel.
@cudaq.kernel
def kernel(qubit_count: int):
    # Allocate our qubits.
    qvector = cudaq.qvector(qubit_count)
    # Place the first qubit in the superposition state.
    h(qvector[0])
    # Loop through the allocated qubits and apply controlled-X,
    # or CNOT, operations between them.
    for qubit in range(qubit_count - 1):
        x.ctrl(qvector[qubit], qvector[qubit + 1])
    # Measure the qubits.
    mz(qvector)
```

This kernel function can accept any number of arguments, allowing for flexibility in the construction of the quantum program. In this case, the `qubit_count` argument allows us to dynamically control the number of qubits allocated to the kernel. As we will see in further examples, we could also use these arguments to control various parameters of the gates themselves, such as rotation angles.