Project List

by Frank Mueller

Fall 2020
Project M1: Compiler for Duke Ion Trap

- Develop a compiler/transpiler
  - from a high-level language (Qiskit, Cirq, C#...) to...
    → ion trap gate-level (direct access)
  1. a simulator (one group) to run @kernel decorated target code
  2. Ion trap gate-level API (“DAX”)
- Support for
  - Loops, functions etc. → classical
  - Quantum-classical interaction, e.g., mapping iterations to qubits, state preparation, sampling/measure
  - Support for multi-kernel (like GPU)
  - Consider scalability with # qubits

```python
@kernel
def SampleQrng(self):
    self.load_ions(2)  # Load ions
    self.initialize()  # Initialize all
    delay(10 * us)
    with parallel:
        self.H(0)  # Hadamard gates
        self.H(1)
    r = self.measure()  # Measure all
    return r[:2]  # Return results

1 operation sampleQrng(): Result {
    Allocate qubits in context
    using ((q0, q1) = (Qubit(), Qubit())) {
        H(q0);  // Hadamard gates
        H(q1);
        let r0 = M(q0);  // Measure qubits
        let r1 = M(q1);
        Reset(q0);  // Reset qubits
        Reset(q1);
        return [r0, r1];  // Return result
    }
}
```
Project M2: IBM Q Noise Assessment

1. Assess noise on IBM Q
   - Track read-out, CNOT, cross-talk noise over time
     - $\frac{1}{2}, 1, 2, 4, 8, 12, 24$ hourly
     - Per qubit/qubit connection
     - Use dedicated mode (need to request access via instructor)
   - Statistically evaluate trends
     - Recommend how frequently to measure which of them

1 group, 2-3 students
Project M3: IBM Q Pulses to Reduce Noise

1. Develop calibration circuit
   — Experiment w/ different pulses for low-level u1/u2/u3 gates
   — Select best pulse to minimize error (noise) per qubit/connect.
   — Measure result in pulse values
     → establishes “new base” for what |0> and |1> means

2. Re-write short circuits manually as u1/u2/u3 w/ new pulses
   — Verify higher fidelity of results
   — Automate standard gate (Pauli, Hadamard...) writing as u1/u2/u3 → Quantum Algorithms for Beginners,
     https://arxiv.org/abs/1804.03719
Project M4: Debugging Execution

- Cannot breakpoint debug
  - measurement destroys superpositions, entanglements
- Best-effort breakpoint debugging
  1. With execution: (group 1)
     - Run up to breakpoint, take 1000 samples
     - Identify Hadamard positioning, entanglements/un-entagle
     - Create synthetic circuit (aka. state preparation) to resemble
       - H/entanglements
       - Histograms of states (for classically-equivalent states)
     - Use circuit as an initializer (aka. state preparation → Qiskit API), run original circuit from breakpoint onward

- Output will only examine a subset of original states
  - Still better than no breakpoint debugging
  - Provide scenarios to demonstrate this point

1 group, 2-3 students
Project M5: Debugging Simulation

- Cannot breakpoint debug
  - measurement destroys superpositions, entanglements
- Best-effort breakpoint debugging

2. With Simulation: (group 2)
  - Same as above but capture all state from simulation
  - Or at least some substate from simulation, esp. for short circuits
  - Then generate initializer (state preparation) as discussed above

- Output will only examine a subset of original states
  - Still better than no breakpoint debugging
  - Provide scenarios to demonstrate this point

1 group, 2-3 students