High Level Programming Languages for Quantum Computation

Birds of a Feather, SC18 – November 2018, Dallas, TX

EuroMPI’19
September 11-13 2019
Zurich, Switzerland

https://eurompi19.inf.ethz.ch
Submit papers by April 15th!
Promising applications of quantum computing

**Quantum Chemistry/Physics**
- Original idea by Feynman – use quantum effects to evaluate quantum effects
- Design catalysts, exotic materials, ...

**Breaking encryption & bitcoin**
- Big hype – destructive impact – single-shot (but big) business case
- Not trivial (requires arithmetic) but possible

**Accelerating heuristical solvers**
- Quadratic speedup can be very powerful!
- Requires much more detailed resource analysis → systems problem

**Quantum machine learning**
- Feynman may argue: “quantum advantage” assumes that circuits cannot be simulated classically → they represent very complex functions that could be of use in ML?
Basing on complex quantum algorithms

Most quantum programs recombine known algorithmic building blocks!

<table>
<thead>
<tr>
<th>Amplitude Amplification</th>
<th>Quantum Fourier Transform</th>
<th>Phase Estimation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Amplify probability of the “right” output</strong></td>
<td><strong>DFT on amplitudes of a quantum state</strong></td>
<td><strong>Measure eigenvalues of a unitary operator</strong></td>
</tr>
<tr>
<td>- Using quantum interference</td>
<td>- $O(n \log n)$ gates for $2^n$ elems</td>
<td>- Used to compute eigenvectors</td>
</tr>
<tr>
<td>- E.g., Grover’s search</td>
<td>- Used in factoring and discrete logarithm</td>
<td>- Used to solve linear systems</td>
</tr>
<tr>
<td>- Often $O(\sqrt{2^n})$ iterations</td>
<td></td>
<td>- Determine eigenvalues in $O\left(\frac{1}{\varepsilon}\right)$ gates</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Quantum Walks</th>
<th>Hamiltonian Simulation</th>
<th>Others</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Speedup mixing times in randomized algorithms</strong></td>
<td><strong>Simulate nature 😊</strong></td>
<td>(not relevant for performance/HPC)</td>
</tr>
<tr>
<td>- Quantum version of random walks</td>
<td>- Exponential speedup (over best known) classical algorithm for quantum effects in physics, chemistry, material science .... problems</td>
<td>- Quantum teleportation</td>
</tr>
<tr>
<td>- Between quadratic and (rarely) exponential speedup</td>
<td></td>
<td>- EPR-pair based proofs/certificates</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Certified random number generation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- ...</td>
</tr>
</tbody>
</table>
Designing an algorithm today?

- Model the computation in Hilbert spaces
  - Algorithmic ideas – develop basic tools

- Design a specific algorithm
  - In terms of building blocks
    - qFFT, amplitude amplification, arithmetic, etc.

- Design a specific program
  - As abstract quantum circuit

- Implement and optimize the program
  - Break into basic gates (from a default gate set)

- Map to a machine
  - Error correction
  - Qubit mapping
How are quantum programs specified?

Classical High Level (e.g., Python)

```python
# a scheduler class, to schedule and run events after a delay
class Scheduler:
    def __init__(self):
        # begin with no events
        self.events = []

    # after the delay, run the function
    def schedule(self, delay, function):
        if delay <= 0:
            # if no delay, run function straight away
            function()
```

Classical Low Level (e.g., inline assembly)

```
08048918 pushl %ebp
08048919 movl %esp,%ebp
0804891b subl $0x4,%esp
0804891d movl $0x0,0xffffffffc(%ebp)
08048925 cmpl $0x63,0xffffffffc(%ebp)
08048929 jle 08048930
0804892b jmp 08048948
0804892d nop
```

Quantum High Level

Scott Pakin (LANL)  Cathy Palmer (Microsoft)  Ali Javadi (IBM)

Damian Steiger (Huawei)  Margaret Martonosi (Princeton)

Quantum Addition