An HPC Systems Guy’s View of Quantum Computing

Torsten Hoefler
ETH Zurich, Switzerland (Professor)
Microsoft Quantum, Redmond (Visiting Researcher)
Who is this guy and what is he doing here?

SPCL
1 professor, 6 scientific staff, 13 PhD students

ETH zürich
6.5k staff, 20k students, focus on research

Applications

Programming Systems

Accelerator Hardware

[Q#] LIQUID

What is a qubit and how do I get one?

“I don’t like it, and I'm sorry I ever had anything to do with it.”
Schrödinger (about the probability interpretation of quantum mechanics)

\[ |\Psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle \quad |\alpha_0|^2 + |\alpha_1|^2 = 1 \]

For example: \[|+\rangle = \frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} |1\rangle \]

One qubit can include a lot of information in \(\alpha_0\) and \(\alpha_1\) but can only sample one bit while losing all

(\text{encoding } n \text{ bits takes } \Omega(n) \text{ operations})

\(n\) qubits live in a vector space of \(2^n\) complex numbers (all combinations + entanglement)

\[ |\Psi_n\rangle = \sum_{i=0..2^n-1} \alpha_i |i\rangle \quad \text{e.g., } |\Psi_2\rangle = \alpha_0 |00\rangle + \alpha_1 |01\rangle + \alpha_2 |10\rangle + \alpha_3 |11\rangle \]
Example: adding $2^n$ numbers in $O(\log n)$ time

Reminder: Classical Adder

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

logarithmic depth

linear work

Quantum Adder

$$|a\rangle = \frac{1}{\sqrt{4}}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$$

$$|b\rangle = \frac{1}{\sqrt{4}}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$$

$$|0\rangle \xrightarrow{H} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

$$|0\rangle \xrightarrow{H} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

$$|0\rangle \xrightarrow{H} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

final adder state
(entangled "probability distribution")

print(a, a+b)

("measure")
We add all $2^n$ numbers in parallel but only recover $n$ classical bits!

A Corollary to Holevo’s Theorem (1973): *at most $n$ classical bits can be extracted from a quantum state with $n$ qubits* even though that system requires $2^n - 1$ complex numbers to be represented!

My corollary: *practical quantum algorithms read a linear-size input and modify an exponential-size quantum state such that the correct (polynomial size) output is likely to be measured.*

Question: Are quantum algorithms good at solving problems where a solution is verifiable efficiently (polynomial time)? Answer: Kind of ☺
So quantum computers can solve NP-complete problems!?

A problem is in NP if a solution can be verified deterministically in polynomial time.

- Even with quantum computing, it seems that $P \neq NP$ (limited by linearity of operators). Quantum is at least as powerful as classic, thus, we do not know!
- New complexity class: **Bounded-error Quantum Polynomial time (BQP)**
  - Quantum version of to **Bounded-error Probabilistic Polynomial time (BPP)**

$PSPACE$ $\supseteq$ $\text{BQP}$ $\supseteq$ $\text{BPP}$ $\supseteq$ $NP$ $\supseteq$ $\text{NP-complete}$

- $NP$-complete – e.g., factoring, discrete logarithm
- $NPI$ – e.g., graph isomorphism

A problem is in $NP$ if a solution can be verified deterministically in polynomial time.
Quantum algorithms are very complex (i.e., weird)

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 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>
How does a quantum computer work?

Qubits are arranged on a (commonly 2D) substrate
Reuse big parts of process technology in microelectronics

Qubits are error prone, need to be highly isolated (major challenge)
Quantum error correction enabled the dream of quantum computers

Operations ("gates") are applied to qubits in place!
As opposed to bits flowing through traditional computers!

Quantum systems are most naturally seen as accelerators
Work in close cooperation with a traditional control circuit

Quantum circuits use predication (no control flow)
Circuit view simplifies reasoning but requires classical envelope

Commonly limited to neighbor interactions between qubits
Limited range, may require swapping across chip

Operations ("gates") have highly varying complexity
Some are literally free (classical tracking), some are very expensive
## Hardware and software architecture for quantum computing

<table>
<thead>
<tr>
<th>Logical layer</th>
<th>qubits</th>
<th>bits</th>
<th>abstraction</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Quantum circuits</td>
<td>Instruction stream + data</td>
<td>Q# programming language</td>
</tr>
</tbody>
</table>

### Gate synthesis
- **Q error correction**: QEC Codes (Steane etc., 1:n mapping)
- **Factory control and qubit routing**

### Physical control
- **Physical quantum state**
- **Analog pulse generators**
- **Qubit control pulses**

### Technical Specifications
- **Logical layer**
  - **qubits**: 300K > 100kW
  - **Gate synthesis**: 77K (liquid nitrogen) < 1 kW
  - **Q error correction**: 4K (liquid helium) < 10 W
  - **Physical control**: < 0.1 K (dilution refrigeration) < 0.01 W

### Abstraction
- **Software (SW)**: Q# programming language
- **Middle Wear (MW)**: Q intermediate representation
- **Hardware (HW)**: Microcoded instructions

### Temperature
- **Classical Logic**:
  - 77K (liquid nitrogen)
  - 4K (liquid helium)
  - < 0.1 K (dilution refrigeration)

- **qubits**
  - 300K > 100kW
  - 77K (liquid nitrogen) < 1 kW
  - 4K (liquid helium) < 10 W
  - < 0.1 K (dilution refrigeration) < 0.01 W

---

*Research Faculty Summit 2018: Systems | Fueling future disruptions*
Full Example: Grover’s search

- Task: find \( x \in D \) for which \( f(x) = y \) (invert \( f(x) \))
- Classical requires \( O(|D|) \) queries
- Quantum requires \( O\left(\sqrt{|D|}\right) \) queries

Q# code

```
operation GroverSearch(n_searchQubits: Int): (Result[])
{
    mutable resultElement = new Result[n_searchQubits];
    using (qubits = Qubit[n_searchQubits]) {
        Allocate[\lceil \log_2 |D| \rceil] qubits
        Allocate \[00, 01, \ldots, 11\] qubits
        ApplyToEachCA(H, qubits); // qubits to uniform superposition
        let n_iterations = Floor(0.25 * PI() * Sqrt(ToDouble(2^n_searchQubits)));
        // Grover iteration
        for (nonce in 1..n_iterations) {
            OracleAND(qubits); // flips phase of desired state
            ApplyToEachCA(H, qubits);
            ApplyToEachCA(X, qubits);
            (Controlled Z)(qubits[1..n_searchQubits-1], qubits[0]);
            ApplyToEachCA(X, qubits);
            ApplyToEachCA(H, qubits);
        }
        set resultElement = MultiM(qubits);
    }
    return (resultElement);
}
```
Quadratic speedup? Grover on a real machine

Performance estimates must be understood to be believed (inspired by Donald Knuth’s “An algorithm must be seen to be believed”)

1. Query complexity model – how algorithms are developed
   • \( T = \frac{\pi}{4\sqrt{2^n}} \) queries (\(|D| = 2^n\) - represented by \( n \) bits)

2. Express (oracle and diffusion operator) as \( n \)-bit unitary
   • Assuming \( O \) \( n \)-bit operations for oracle!
   • \( T = O \left( \frac{\pi}{4\sqrt{2^n}} \right) n \)-bit operations - \( T_t = \left[ \frac{\pi}{4\sqrt{2^n}} \right] \)

3. Decompose unitary into two-bit (+arbitrary rotation) gates
   • \( T = O_2 \left( \frac{\pi}{4\sqrt{2^n}} \right) \cdot 2(n - 1) \) elementary operations - \( T_t = \left[ \frac{\pi}{4\sqrt{2^n}} \right] \cdot 4(n - 1) \)

4. Design approximate implementations in discrete gate set (using HTHT...)
   • \( T = O_2 \left( \frac{\pi}{4\sqrt{2^n}} \right) \cdot 2(n - 1) \) discrete T gate operations - \( T_t = \left[ \frac{\pi}{4\sqrt{2^n}} \right] \cdot 48(n - 1) \)

5. Mapping to real hardware (swaps and teleport)
   • Not to simple to model, depends on oracle – potentially \( \Theta(\sqrt{2^n}) \) slowdown

6. Quantum error correction
   • Not so simple, depends on quality of physical bits and circuit depth, huge constant slowdown
Quadradic speedup? Grover on a real machine

Performance estimates must be understood to be believed (inspired by Donald Knuth’s “An algorithm must be seen to be believed”)

1. Query complexity model – how algorithms are developed
   • \( T = \frac{\pi}{4} \sqrt{2^n} \) queries (|D| = \( 2^n \) represented by n bits)

2. Express (oracle and diffusion operator) as n-bit unitary
   • Assuming \( O \) n-bit operations for oracle!
   • \( T = \frac{n}{2} \pi \sqrt{2^n} \) n-bit operations - \( T_t = \frac{n}{2} \pi \sqrt{2^n} \)

3. Decompose unitary into two-bit (+arbitrary rotation) gates
   • \( T = \frac{n}{2} \pi \sqrt{2^n} \cdot 2 \cdot (n - 1) \) elementary operations
   • \( T_t = \frac{n}{2} \pi \sqrt{2^n} \cdot 4 \cdot (n - 1) \)

4. Design approximate implementations in discrete gate set (using HTHT...)
   • \( T = \frac{n}{2} \pi \sqrt{2^n} \cdot 2 \cdot (n - 1) \) discrete T gate operations
   • \( T_t = \frac{n}{2} \pi \sqrt{2^n} \cdot 4 \cdot (n - 1) \)

5. Mapping to real hardware (swaps and teleport)
   • Not to simple to model, depends on oracle – probably \( \Theta \) 2^n slowdown

6. Quantum error correction
   • Not so simple, depends on quality of physical bits and circuit depth, huge constant slowdown

Quantum computer with logical error rates \( \leq 10^{-24} \) and gate times of \( 10^{-6} \)s vs. classical at 1 teraop/s.

<table>
<thead>
<tr>
<th>k</th>
<th>#gates</th>
<th>T</th>
<th>Clifford</th>
<th>T</th>
<th>depth</th>
<th>#qubits</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>1.19</td>
<td>2^86</td>
<td>1.55</td>
<td>2^86</td>
<td>1.06</td>
<td>2^80</td>
</tr>
<tr>
<td></td>
<td>1.16</td>
<td>2^81</td>
<td>1.16</td>
<td>2^81</td>
<td>1.16</td>
<td>2^81</td>
</tr>
<tr>
<td>192</td>
<td>1.81</td>
<td>2^118</td>
<td>1.17</td>
<td>2^119</td>
<td>1.21</td>
<td>2^112</td>
</tr>
<tr>
<td></td>
<td>1.33</td>
<td>2^113</td>
<td>1.33</td>
<td>2^113</td>
<td>1.33</td>
<td>2^113</td>
</tr>
<tr>
<td>256</td>
<td>1.41</td>
<td>2^151</td>
<td>1.83</td>
<td>2^151</td>
<td>1.44</td>
<td>2^144</td>
</tr>
<tr>
<td></td>
<td>1.57</td>
<td>2^145</td>
<td>1.57</td>
<td>2^145</td>
<td>1.57</td>
<td>2^145</td>
</tr>
</tbody>
</table>

Table 5. Quantum resource estimates for Grover’s algorithm to attack AES-k, where k \( \in \{128, 192, 256\} \).

Real applications?

**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?
Thanks!

• Special thanks to Matthias Troyer and Doug Carmean

• Thanks to: Thomas Haener, Damian Steiger, Martin Roetteler, Nathan Wiebe, Mike Upton, Bettina Heim, Vadym Kliuchnikov, Jeongwan Haah, Dave Wecker, Krysta Svore

• And the whole MSFT Quantum / QuArC team!

All used images belong to the respective owners – source list in appendix