Toward optimizing compilers for quantum computers

Jan 17, 2019
PEQUAN seminar

Sylvain Collange
Inria, Univ Rennes, CNRS, IRISA
sylvain.collange@inria.fr
Why Quantum Computing Today?

- Already an established research topic since 1990's
  - In theoretical computer science
  - In applied quantum physics
- Usual stance in applied CS: *I'll believe it when I'll see it.*
- Today: no excuse, we got hardware!
  - IBM: **open access**
    16-bit quantum computer,
    20-qubit in limited access,
    50-qubit prototype
  - Rigetti
    19-qubit in limited access
  - Google, Microsoft, Intel…
    various prototypes
    up to 72-qubit

- Enables **experimental** computer science
- Opportunity for computer architecture and compiler research
Babbage: “It seems to me probable that a long period must elapse before the demands of science will exceed this limit.”

As of 2018

- 50-digit (~170-bit) numbers still considered ludicrous precision
- Complete Analytical Engine has yet to be built
Welcome to the NISQ era

John Preskill keynote: Quantum computing in the NISQ era and beyond

Today: we have real quantum hardware
- But too few, noisy, qubits to implement 1990's algorithms
- A few near-term applications: quantum chemistry simulation

Crossroads for the quantum computing field
- Success → sustained investments toward more ambitious applications
- Failure → quantum computing winter for the next 20-30 years
Agenda

- Introduction to the programming model
  - Logical qubits and quantum gates

- Compiling quantum circuits
  - Allocating logical qubits on physical qubits
What is so special about quantum?

- Example: Young's double-slit experiment
  - Each photon behaves as a wave: goes through both holes and interferes with itself
  - Idea: craft quantum experiments to perform computations
  - Quantum computing approach
    - Compute on superposed states
    - Exploit interference to select useful information
    - Measure results to infer statistical distribution
Computing abstraction: Quantum circuit

- Like classical circuit or dataflow graph, except:
  - Operates on qubits
  - Reversible: no creation, destruction, nor duplication of qubits
  - Starts by initialization, ends by measurement
Basic data-type: the qubit

- Superposition of states: $\alpha|0\rangle + \beta|1\rangle$ with $\alpha, \beta \in \mathbb{C}$, $|\alpha|^2 + |\beta|^2 = 1$

  - Representation as vector in basis ($|0\rangle$, $|1\rangle$): \[
  \begin{pmatrix}
  \alpha \\
  \beta
  \end{pmatrix}
  \]

- We can visualize possible states on the surface of a sphere
Multiple qubits

- State space: exponential number of dimensions
  - $n$ classical bits encode **one** of $2^n$ states: space is $\{0,1\}^n$
  - $n$ qubits encode a **superposition** of $2^n$ states: space is $\mathbb{C}^{2^n}$ (normalized)

- From independent qubits
  - Tensor product of individual states
    $$a|0\rangle + b|1\rangle \quad \text{and} \quad c|0\rangle + d|1\rangle$$
    $$(a|0\rangle + b|1\rangle) \otimes (c|0\rangle + d|1\rangle) = ac|00\rangle + ad|01\rangle + bc|10\rangle + bd|11\rangle$$

- State may not be separable: qubits are in an **entangled** state
  - e.g.
    $$|\rangle \quad \text{and} \quad |\rangle$$
    $$1/\sqrt{2} |00\rangle + 1/\sqrt{2} |11\rangle$$

No $a,b,c,d$ such that $ac=bd=1/\sqrt{2}$ and $ad=bc=0$

- Need to consider group of entangled qubits as a whole

- Visualization: $2^{2n}$-dimension hypersphere? 😊
Measurement turns a qubit into a bit

- Measuring \( \alpha |0\rangle + \beta |1\rangle \) gives:
  - 0 with probability \( |\alpha|^2 \)
  - 1 with probability \( |\beta|^2 \)

- Destructive operation
  - State space of the system projected to \( \mathbb{C}^{2^n-1} \)
  - No information on sign / complex phase
  - Random: need to repeat to infer distribution

\[
\begin{align*}
\frac{1}{\sqrt{2}} |0\rangle - \frac{1}{\sqrt{2}} |1\rangle & \quad \text{projection along Z axis}
\end{align*}
\]

\[
\begin{align*}
\frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} |1\rangle & \quad p = \frac{1}{2}
\end{align*}
\]
Operation: single-qubit gate

Quantum gates as mul by unitary matrices

- Correspond to rotations on the sphere
- e.g. X gate
  - flip along X axis
  - maps $|0\rangle$ to $|1\rangle$ and $|1\rangle$ to $|0\rangle$
  - “equivalent” of classical NOT
Operation: single-qubit gate

Quantum gates as multiplication by unitary matrices

- Correspond to rotations on the sphere
- e.g. X gate
  - flip along X axis
  - maps $|0\rangle$ to $|1\rangle$ and $|1\rangle$ to $|0\rangle$
  - “equivalent” of classical NOT

- e.g. Hadamard-Walsh gate
  - maps $|0\rangle$ to $\frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} |1\rangle$
    and $|1\rangle$ to $\frac{1}{\sqrt{2}} |0\rangle - \frac{1}{\sqrt{2}} |1\rangle$

- Any single-qubit gate can be decomposed into a sequence of X and Z axis rotations
Multi-qubit gate: Controlled NOT

- **CNOT or Controlled-X:** analog of classical XOR

\[ a|00\rangle + b|01\rangle + c|10\rangle + d|11\rangle \]

\[ \text{CNOT gate} \]

\[ a|00\rangle + b|01\rangle + d|10\rangle + c|11\rangle \]

“Flips second qubit when first qubit is \(|1\rangle\)”

- As a way to entangle qubits

\[ \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \]

\[ \text{Z gate} \]

\[ \text{H gate} \]

\[ \text{X gate} \]

\[ \text{H gate} \]

\[ \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|10\rangle \]

\[ \text{Z gate} \]

\[ \text{H gate} \]

\[ \text{H gate} \]

\[ \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle \]

- As a building block to make arbitrary controlled gates

\[ \text{e.g.} \]

\[ \text{Z gate} = \text{H gate} \times \text{H gate} \]

\[ \text{and} \]

\[ \text{H gate} \times \text{H gate} \]

\[ \text{Z gate} \]

\[ \text{H gate} \times \text{H gate} \]

\[ \text{H gate} \]

\[ \text{H gate} \]

13
Agenda

- Introduction to the programming model
  - Logical qubits and quantum gates

- Compiling quantum circuits
  - Allocating logical qubits on physical qubits
Compilers for quantum computing

- Existing and near-future architectures:
  - 10s to 100 qubits
  - No error correction
  - Low-level constraints on circuits: set of gates, qubit connectivity

- Need compilers of circuits down to low-level gates
  - Many differences from classical compilers
Focus: the qubit allocation phase

- Map logical qubits to physical qubits
  - Need to meet hardware constraints: connectivity between physical qubits
  - Transform circuit to fit on given quantum computer

- Minimize runtime and gate count to **minimize noise**

Software: circuit on logical qubits

Hardware: physical qubits

Joint work with Marcos Yukio Siraichi, Vinícius Fernandes dos Santos and Fernando Magno Quintão Pereira, *DCC, UFMG, Brazil*
Input: reversible quantum circuits described at gate level

- Between initialization and measurement: unitary gates only
- After decomposition into single-qubit and CNOT gates
- Expressed in QASM language

```qasm
qreg l[2];
creg c[2];
x l[0];
h l[0];
cx l[0] l[1];
t l[1];
measure l[0] -> c[0];
measure l[1] -> c[1];
```
Limited-connectivity quantum computer

Target: superconducting qubit based quantum computers

- Constraints on which qubits are allowed to interact
- e.g. IBM QX2, 5 qubits
- e.g. IBM QX5, 16 qubits
Qubit assignment is Subgraph Isomorphism

Can we label logical qubits with physical qubits so that all gates obey machine connectivity constraints?

- Known as the Subgraph Isomorphism problem
- “Easy part” of qubit allocation
- Already NP-Complete

In practice, most circuits will need transformations to “fit” the connectivity graph
Circuit transformation primitives

Transformation

- **CNOT reversal**
  - a
  - b

- **Bridge**
  - a
  - b
  - c

- **Swap**
  - a
  - b

Effect on dependency graph (assuming no other dependency)

- **Change mapping!**
  - a
  - b
  - c
1. Compute maximal isomorphic partitions

- Break circuit into solvable instances of subgraph isomorphism
  - Maximal: adding one dependency makes it unsolvable
- Approximated with bounded exhaustive search
  - For each partition, build collection of candidate mappings

**Connectivity graph**

```
  a --- c --- e  
  |     |     |  
  b --- d --- f
```

**Example candidate mappings**

```
  1 --- 2  
  |     |  
  3 --- 5 --- 4
```

**Circuit**

```
  1 --- 2  
  |     |  
  1 --- 3 --- 4
  |     |  
  1 --- 2
```
2. Choose qubit mappings, add swaps

Select one mapping in each partition

- Goal: minimize total number of swaps
- Equivalent to Token Swapping problem (NP hard)
- Use 4-approximation algorithm proposed in 2016
Comparison with other approaches

- Cost (lower is better)

![Comparison of algorithms](image)

- Proposed algorithm

RevLib Benchmarks
Conclusion: compiler optimization for quantum circuits

An entire domain to explore

- Qubit allocation
  - Seek run-time vs. accuracy tradeoffs, optimize for fidelity
  - Specialize for regular quantum computer structures
  - Take advantage of quantum circuit properties: spacial, temporal locality

- Mapping high-level gates to hardware-supported gates
  - High-level gate implementation: accuracy/cost tradeoffs
  - Selecting gate sequences: use degree of freedom on relative phase

- Time/space tradeoffs
  - Adapt number of helper qubits to resource availability

- Formalization
  - Which semantics for quantum programs and quantum computers?
  - Which intermediate representation for quantum circuits?