Partial Compilation of Variational Algorithms for Noisy Intermediate-Scale Quantum Machines

Pranav Gokhale\textsuperscript{1} \quad Yongshan Ding\textsuperscript{1} \quad Thomas Propson\textsuperscript{2}
\quad Christopher Winkler\textsuperscript{1} \quad Nelson Yeung\textsuperscript{2} \quad Yunong Shi\textsuperscript{2}
\quad David I. Schuster\textsuperscript{2} \quad Henry Hoffmann\textsuperscript{1} \quad Frederic T. Chong\textsuperscript{1}

\textsuperscript{1}Department of Computer Science
\quad University of Chicago
\textsuperscript{2}Department of Physics
\quad University of Chicago

MICRO-52, October 2019
Outline

- Background: Quantum Compilation
- Variational Quantum Algorithms
- Partial Compilation
  - Strict
  - Flexible
- Results
- Conclusions / Future Work
• 5x improvement in quantum runtime
• Qubits decay exponentially with time
• How? Compilation to pulses
module QFT(qbit x[]) {
    int i, j;
    double angle = 0.0;
    for (i=0; i<n; i++) {
        H(x[i]);
        for (j=i+1; j<n; j++) {
            angle = angle_R[j];
            cRz(x[j], x[i], angle);
        }
        x q[2];
        barrier q;
        h q[0];
        cu1(pi/2) q[1], q[0];
        h q[1];
        cu1(pi/4) q[2], q[0];
        cu1(pi/2) q[2], q[1];
        h q[2];
        cu1(pi/8) q[3], q[0];
        cu1(pi/4) q[3], q[1];
        cu1(pi/2) q[3], q[2];
        h q[3];
        measure q -> c;
    }
}
Background

**Quantum Programs**

<table>
<thead>
<tr>
<th>Standard Gate Based Compilation</th>
<th>Pulse-Based Compilation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Compiler Frontend</strong></td>
<td><strong>Compiler Frontend</strong></td>
</tr>
<tr>
<td>QASM</td>
<td>QASM</td>
</tr>
<tr>
<td>Compiler Backend</td>
<td>Commutativity Detection</td>
</tr>
<tr>
<td>QISA</td>
<td>Aggregation</td>
</tr>
<tr>
<td>Assembler</td>
<td>Aggregated Instructions</td>
</tr>
<tr>
<td>Control Pulses</td>
<td>Optimal Control</td>
</tr>
<tr>
<td><strong>Optimized Control Pulses</strong></td>
<td></td>
</tr>
</tbody>
</table>

Aggregation + Optimal Control (GRAPE)
GRAPE (GRAdient-ascent Pulse Engineering)

Gokhale et al.

Partial Compilation
In sum: ASPLOS paper demonstrated how to get large pulse speedups, but takes a long time to compile.
• Originally quantum algorithms are fully specified at compile time
  • Shor Factoring, Grover Search, etc.
• But, hardware requirements are untenable for Noisy Intermediate-Scale Quantum (NISQ) devices
• Variational quantum algorithms match NISQ hardware
Variational Quantum Algorithms

- Quantum program is not fully specified at compile time
- Program is executed multiple times, varying choices of unspecified parameters
- Classical co-processor optimizes choice of parameters, based on history of past executions
Variational Quantum Algorithms

Variational Algorithm

Quantum

Classical

Compile

Pulse optimization

New trial parameters

Quantum hardware

"Parameterized circuit"

Input parameters: $\hat{\theta}$

Evaluate: $E[\hat{\theta}]$

Classical hardware

"Gradient descent"

Next guess: $\hat{\theta}$

Search for $E_{\text{min}}$

Output $(\hat{\theta}, E_{\text{min}}[\hat{\theta}])$ after sufficient iterations.

Loops 1000s of times

"Gokhale et al.

Partial Compilation"
• New challenge for pulse-level compilation
• Need to compile thousands of programs
• And compilation latency delays time-to-solution
Partial Compilation

• Natural response to a partially specified program
• Two flavors:
  • Strict partial compilation. Pre-compile parts of the program.
  • Flexible partial compilation. Pre-compute good hyperparameters to speed up future compilation.
Strict Partial Compilation

Gokhale et al.
Strict partial compilation *pre-compiles* Fixed blocks—no runtime latency.
- GRAPE does best with deep (wide) slices
- Strict is limited by interspersed parametrization-dependent gates
- Instead, consider slices parametrized by exactly 1 parameter
- Can’t pre-compile, but …
- Can find good hyperparameters
Flexible Partial Compilation

\[ R_z(\theta_1) \]

\[ R_z(\theta_2) \]

\[ R_z(\theta_3) \]
Flexible Partial Compilation

Gokhale et al.

Partial Compilation
Flexible Partial Compilation

Parameter Monotonicity

\( R_Z(\theta_1) \)

\( R_Z(\theta_2) \)

\( R_Z(\theta_3) \)
Flexible Partial Compilation

\[ R_z(\theta_1) \]

\[ R_z(\theta_2) \]

\[ R_z(\theta_3) \]

\( \theta_1 \)-dependent

\( \theta_2 \)-dependent

\( \theta_3 \)-dependent
Results: VQE

Gokhale et al.

Partial Compilation
Results: QAOA

![Graph showing Quantum Runtime Speedup](image)

- **3-Regular N=6**
- **Erdos-Renyi N=6**
- **3-Regular N=8**
- **Erdos-Renyi N=8**

**Legend:**
- Gate
- Strict
- Flexible
- GRAPE
Results: Side-by-Side

**VQE**
- BeH₂ (6 Qubits): 5.3 μs
- NaH (8 Qubits): 5.5 μs
- H₂O (10 Qubits): 33.8 μs

**QAOA**
- 3-Regular N=6
- Erdos–Renyi N=6
- 3-Regular N=8
- Erdos–Renyi N=8

Quantum Runtime Speedup

Gokhale et al. Partial Compilation
Results: Side-by-Side

VQE

QAOA

Gokhale et al.  Partial Compilation
Results: Compilation Latency Reduction

Strict has 0 compilation time during runtime. For flexible:

<table>
<thead>
<tr>
<th></th>
<th>VQE</th>
<th>QAOA</th>
</tr>
</thead>
<tbody>
<tr>
<td>BeH₂</td>
<td>17163s</td>
<td>12786 s</td>
</tr>
<tr>
<td>NaH</td>
<td>12387s</td>
<td>23718 s</td>
</tr>
<tr>
<td>H₂O</td>
<td>19065s</td>
<td>11645 s</td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3Reg N=6</td>
<td>305 s</td>
<td>159 s</td>
</tr>
<tr>
<td>3Reg N=8</td>
<td>1057 s</td>
<td>289 s</td>
</tr>
<tr>
<td>Erdos N=6</td>
<td>1261 s</td>
<td>263 s</td>
</tr>
<tr>
<td>Erdos N=8</td>
<td>1258 s</td>
<td>1258 s</td>
</tr>
</tbody>
</table>

Gokhale et al.  Partial Compilation
Conclusions / Future Work

• 2x pulse speedups with minimal compilation latency
  • Strict has no latency
  • Flexible partial compilation has latency, but 10-80x lower than standard GRAPE.

• We propose experimental realizations, e.g. via OpenPulse

• Also a need for better (faster and smarter) GRAPE implementations
Partial Compilation of Variational Algorithms for Noisy Intermediate-Scale Quantum Machines

Pranav Gokhale, Yongshan Ding, Thomas Propson, Christopher Winkler, Nelson Leung, Yunong Shi, David I. Schuster, Henry Hoffmann, Frederic T. Chong

INTRO and METHODS

• Variational quantum circuits are costly in run-time (gates) or in compile time (pulses).

RESULTS

• > 2x speedups for VQE and QAOA with minimal compilation latency.

DISCUSSION

• Success probability decays exponentially in pulse runtime — speedups are critical.
• Results persist for realistic pulses. Experimental efforts ongoing.

Partial compilation enables 2x pulse speedups for quantum circuits, with reduced compilation latency.