Compilação e Execução de código da linguagem QML no Computador Quântico da IBM

João Gabriel da Cunha Schittler, Juliana Kaizer Vizzotto

[1]Curso de Ciência da Computação – Universidade Federal de Santa Maria (UFSM)
Caixa Postal – Santa Maria – RS – Brasil

{jgschittler,juvizzotto}@inf.ufsm.br

Abstract. *This paper descibes an ongoing project that aims to implement a compiler for the quantum programing language QML that generates quantum circuits, and then execute these compiled circuits on a IBM quantum computer using the Qiskit package.*

Resumo. *Esse artigo descreve um projeto em andamento que propõe implementar um compilador para circuitos quânticos para a linguagem de programação quântica QML e então executar os circuitos compilados no computador quântico da IBM (*IBM Quantum Computer*) usando o pacote Qiskit.*

1. Introdução

A computação quântica tem sua origem no trabalho de Richard Feynman [Feynman 1982], onde o autor trata questões sobre simular sistemas físicos quânticos com a utilização de computadores clássicos. Desta maneira, Feynman propôs utilizar fenômenos da mecânica quântica, como superposição e emaranhamento, para resolver problemas mais rapidamente que na computação clássica. Essa questão levantada por Feynman levou à algumas propostas para um modelo quântico de computação, mas a realização de que realmente pode-se ter algum ganho veio somente alguns anos depois com os resultados de Grover[Grover 1996] e Shor [Shor 1997]. Grover propôs um algoritmo quântico para executar busca em registros desordenados, com um ganho quadrático na complexidade temporal comparando com qualquer outro algoritmo clássico. O trabalho de Shor definiu um algoritmo quântico de tempo polinomial para fatorar números inteiros. Para esse problema somente se conhece algoritmos clássicos com tempo exponencial. Pode-se dizer que esses foram os primeiros resultados que geraram um grande interesse de pesquisadores da ciência de computação.

Principalmente com o objetivo de investigar a área da programação quântica e proporcionar o desenvolvimento e estudo de algoritmos quânticos, as linguagens de programação quânticas de alto nível têm sido desenvolvidas nos últimos anos. A linguagem funcional quântica QML [Altenkirch and Grattage 2005] é um exemplo. QML é baseada nas construções de alto nível das linguagens funcionais convencionais e integra computações quânticas reversíveis e irreversíveis, usando lógica linear de primeira ordem para os *weakenings*. Os programas *estritos* não apresentam decoerência quântica e consequentemente preservam superposições e emaranhamento, que são essenciais para o paralelismo quântico. Além disso, os autores apresentam uma maneira natural de compilar os termos funcionais da linguagem em circuitos quânticos. Em [Grattage and Altenkirch 2005] os autores apresentam a implementação de um compilador em Haskell para QML.

O computador quântico da International Business Machines (IBM) e sua plataforma IBM Quantum Experience (IBM-QE) [IBM] proporcionam o uso de um computador quântico em nuvem que pode ser acessado de forma remota por qualquer um.

Com objetivo de investigar os potenciais de linguagens de programação quânticas de alto nível e algoritmos quânticos e também estimular esse ramo de pesquisa na área de Ciência da Computação, o presente trabalho propõe implementar um compilador para QML considerando sua semântica de circuitos quânticos, que possa ser executado no computador quântico da IBM, através da integração com o pacote Qiskit [Qis].

2. Fundamentação Teórica

2.1. Computação Quântica

Computação quântica (CQ)[Nielsen and Chuang 2011], considerando um ponto de vista algorítmico, apresenta um novo modelo de computação, o qual incorpora novas maneiras de projetar e estruturar algoritmos utilizando as características da física quântica. A unidade básica de informação clássica computável é o bit, um sistema físico clássico binário. Em computação quântica, a unidade básica de informação é representada pelo bit quântico ou qubit, um sistema físico quântico binário. O qubit é comumente representado como uma superposição de estados, através da notação *braket*[1] de Dirac: $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$.

Os valores associados às bases α e β representam as amplitudes de probabilidade relacionadas a cada base do qubit ($|0\rangle$ e $|1\rangle$). O qubit pode ser definido como um vetor em um espaço vetorial complexo (espaço de Hilbert), tal que $|\alpha|^2 + |\beta|^2 = 1$.

Formalmente, a combinação de dois ou mais estados quânticos pode ser obtida usando uma operação de produto tensorial ($\otimes$). Se $q = \alpha|0\rangle + \beta|1\rangle$ e $p = \gamma|0\rangle + \delta|1\rangle$ são dois qubits não relacionados. Ao aplicar o produto tensorial temos $q \otimes p = \alpha\gamma|00\rangle + \alpha\delta|01\rangle + \beta\gamma|10\rangle + \beta\delta|11\rangle$.

O processamento de informação quântica é realizado por operadores, que possibilitam a evolução de um sistema quântico. Ele define que um sistema quântico isolado no estado $|\phi\rangle_1$ evolui para $|\phi\rangle_2$ através da aplicação de uma operação unitária $|\phi\rangle_2 = U|\phi\rangle_1$. As operações quânticas tem como característica serem reversíveis, quando se conhece as operações que foram aplicadas ao qubit, é possível retornar ao seu estado inicial.

Outro tipo de operação sobre os bits quânticos é a *medição*, esta uma operação não reversível. Ela é obtida com um colapso na função de onda de um sistema quântico. Dessa forma ao aplicar a medição do valor 0 em um sistema quântico da forma $\alpha|0\rangle + \beta|1\rangle$ obtemos $|0\rangle$ com $|\alpha|^2$ de probabilidade associada.

2.2. Linguagem QML

Uma das maiores dificuldades no desenvolvimento de novos algoritmos quânticos, principalmente considerando pesquisadores da área da Ciência da Computação, é o quão baixo nível esses programas são representados, normalmente descritos como circuitos quânticos. A linguagem de programação funcional quântica

[1] O nome *braket* surge através de uma convenção em que um vetor de coluna é chamado "ket" e sua notação é demonstrada por $|\ \rangle$ e um vetor de linhas é chamado "bra" e tem como notação $\langle\ |$.

QML[Altenkirch and Grattage 2005] traz uma abordagem de alto nível, para auxiliar o desenvolvimento desses algoritmos, contendo estruturas de controle de alto nível como variáveis tipadas, funções e as *keywords* **let**,**if-then-else**.

A semântica dessa linguagem pode ser interpretada como uma sequência de FQCs (*finite quantum computations*), que são representadas com a seguinte quíntupla $\mathbf{FQC} = \{a, h, b, g, \phi\}$, onde a representa o número de qbits da entrada, h representa o tamanho da heap inicial, b representa o número de qbits da saída, g representa o tamanho de lixo necessário e ϕ representa uma operação unária reversível.

2.3. IBM Quantum

A empresa IBM criou um serviço chamado de IBM Quantum Experience [IBM], que, entre outras funções, fornece uma interface gráfica para a criação e execução de circuitos quânticos, e disponibiliza uma interface de comunicação via linguagem de programação Python, chamada de Qiskit [Qis], para executar circuitos programados em código. Além disso, o IBM-QE disponibiliza tutoriais e guias de conceitos sobre CQ, que deixa a execução de algoritmos quânticos bem mais acessível.

Os componentes disponíveis para a criação dos circuitos na interface gráfica e pela biblioteca Qiskit são: portas quânticas, como Hadamard, portas lógicas clássicas e outras operações, como medição de qbit.

O pacote Qiskit permite que o programador declare circuitos quânticos, as portas quânticas dos circuitos, registradores clássicos e quânticos, e a interface de comunicação com os serviços da IBM. No código, a ordem em que o programador declara as operações do circuito é a ordem em que elas vão ser executadas no IBM-QE. Após declarar o circuito que vai ser executado, ele pode ser enviado a um computador quântico da IBM.

3. Implementação de um Compilador para a Linguagem de Programação Quântica QML e Execução no IBM-QE

O presente trabalho propõe implementar um compilador para a linguagem de programação quântica QML, considerando sua semântica de circuitos quânticos, que possam ser executados em computadores quânticos da IBM, através da integração com a biblioteca Qiskit.

A compilação do código da linguagem QML vai passar pelos passos padrão de um compilador: análise léxica, sintática e semântica e geração de código. A linguagem de programação que será utilizada para implementação do compilador é a linguagem C++. Pretende-se utilizar a ferramenta LEX para a análise léxica e a ferramenta YACC para a análise sintática e semântica.

O processo da compilação, de forma geral, vai funcionar da seguinte maneira:

- Similar ao que acontece na linguagem QML, cada procedimento/função vai ser expressado em código por uma estrutura que vai conter as informações da quíntupla FQC. Depois que todas as operações do procedimento forem lidas, um circuito representando essa função será criado.
- Quando uma função é chamada dentro de outra função ou pela main, o circuito dessa função é colocado no lugar dessa chamada, e a indexação dos qubits desse

procedimento se ajusta de acordo com os qubits do contexto que chamou o procedimento, de forma que não existam conflitos.
- Depois que todas as funções forem compiladas, a main do programa vai gerar o circuito final que vai ser escrito no arquivo de saída do compilador.

Está sendo estudado alguma maneira de reutilizar qubits auxiliares de procedimentos, de forma que não causem "de-coerência" quântica com os outros qubits, para que os programas compilados usem a menor quantidade de qubits possível.

Os circuitos quânticos gerados pelo compilador vão estar em um formato que pode ser facilmente lido pelo programa que fará a comunicação com o IBM Quantum para então ser executado. O formato do arquivo deve conter os seguintes componentes:

- Um *Header* inicial, contendo informações que precisam ser lidas antes do circuito, como o número de qubits total do circuito.
- A lista de operações do circuito, juntamente com os qbits envolvidos em cada uma.

Por exemplo, um arquivo gerado pelo compilador contendo um programa quântico, seria como abaixo:

- 2
- Had 1
- NOT 2
- CNOT 0 1
- Measure 0
- Measure 1

Na sequência, a arquitetura proposta conterá um programa implementado na linguagem Python, que fará a leitura desse arquivo gerado, montará o circuito quântico e realizará a comunicação com o IBM-QE, usando o pacote Qiskit.

A proposta descrita neste artigo contempla o trabalho de conclusão do curso de Ciência da Computação da Universidade Federal de Santa Maria (UFSM) do primeiro autor. O trabalho está em fase de revisão bibliográfica e delineamento da arquitetura. A implementação da arquitetura aqui proposta ocorrerá nos meses de outubro de 2021 até fevereiro de 2022.

References

Ibm quantum website. `https://quantum-computing.ibm.com/`. acessado em 24/09/2021.

Qiskit website. `https://qiskit.org/`. acessado em 24/09/2021.

Altenkirch, T. and Grattage, J. (2005). A functional quantum programming language. In *20th Annual IEEE Symposium on Logic in Computer Science*.

Feynman, R. P. (1982). Simulating physics with computers. 21(6):467–488.

Grattage, J. and Altenkirch, T. (2005). A compiler for a functional quantum programming language. Manuscript.

Grover, L. K. (1996). A fast quantum mechanical algorithm for database search. In *Proceedings of STOC-1996*, pages 212–219. ACM.

Nielsen, M. A. and Chuang, I. L. (2011). *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, USA, 10th edition.

Shor, P. W. (1997). Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509.