Proceeding Series of the Brazilian Society of Applied and Computational Mathematics, Vol. 2, N. 1, 2014.
Trabalho apresentado no CMAC-Sul, Curitiba-PR, 2014.

Abordagem Paralela para Simulação Quântica no Modelo de Máquina Geométrica Quântica

Murilo Schmalfuss[*], **Anderson Ávila**[†], **Renata Reiser**, **Maurício Pilla**,
Centro de Desenvolvimento Tecnológico, UFPel,
96010-610, Pelotas, RS
E-mail: mfschmalfuss, abdavila, reiser, pilla@inf.ufpel.edu.br

Adriano K. Maron
Department of Computer Science
University of Pittsburgh
Sennott Square, Pittsburgh/PA, USA
E-mail: akk48@pitt.edu.

RESUMO

A Computação Quântica (*CQ*) [7] segue atingindo novos marcos rumo à construção de computadores quânticos. Apesar de todos os esforços, vários desafios técnicos limitam os sistemas atuais a alguns bits quânticos [10]. Pela indisponibilidade de hardware quântico, o estudo e desenvolvimento de aplicações na *CQ* usualmente é feito estritamente pela especificação matemática das computações ou por meio de ferramentas de simulação. Este último caracteriza a abordagem mais prática, entretanto a complexidade computacional associada a simulação de sistemas quânticos a partir de computadores clássicos limita o tamanho dos sistemas a serem simulados.

O projeto no qual este trabalho está inserido busca consolidar a integração de vários esforços de pesquisa, com destaque para: (*i*) Distribuição das computações em *cluster* [2]; (*ii*)Simulação quântica através de *GPUs* [5]; (*iii*) Simulação quântica através de *CPUs multicore*.

Atualmente situado sob o contexto do ambiente de simulação quântica *VPE-qGM* (*Visual Programming Environment for the Quantum Geometric Machine Model*), este trabalho tem o objetivo de estabelecer o suporte à aceleração da biblioteca de execução do ambiente através de processadores *multicore*, beneficiando-se dos recursos providos pela biblioteca *OpenMP* [3]. Consolida-se assim a biblioteca *qGM$_C$-Analyzer*, com a implementação, desenvolvimento e validação de algoritmo para simulação quântica em arquiteturas *multicore*, cuja modelagem foi introduzida em [9].

O ambiente *VPE-qGM*, fundamentado no modelo de processos *qGM* (*Quantum Geometric Machine Model*) [8], é constituído de construtores para modelagem e simulação gráfica de aplicações quânticas. De acordo com o modelo *qGM*, a noção de portas quânticas pode ser substituída pelo conceito de sincronização de processos elementares (*PEs*).

No ambiente *VPE-qGM*, o *PE* é um elemento estruturado por três atributos: (*i*) *Ação*: Corresponde às transformações quânticas aplicadas a diferentes *qubits* em um mesmo instante de tempo; (*ii*) *Parâmetros*: Contém dados auxiliares associados à definição das transformações quânticas; (*iii*) *Posição*: Posição de escrita em um espaço de memória global e compartilhada, na qual é armazenado o resultado calculado pelo *PE*.

Neste contexto, uma transformação quântica, aplicada a *N qubits*, pode ser modelada pela sincronização de 2^N *PEs*, cujas parametrizações satisfazem as condições equivalentes à definição dos vetores componentes da matriz (transformação unitária ou de medida) associada [6].

[*]Bolsista de Iniciação Científica PIBIC/CNPq
[†]Bolsista de PROBIC/FAPERGS

Assim, durante a simulação, ocorre a execução (sequencial ou síncrona) dos *PEs*, os quais têm suas correspondentes computações efetuadas pela biblioteca *qGM-Analyzer*, manipulando os dados presentes nas posições de memória e simulando o comportamento de um sistema quântico.

A biblioteca de execução dos *PEs*, denominada *qGM-Analyzer*, implementa otimizações que controlam o aumento exponencial dos vetores componentes das matrizes de definição do operador de múltiplos *qubits*, conforme introduzido em [4]. Os resultados relacionados comprovam a redução no consumo de memória durante a simulação, suportando algoritmos com 11 *qubits*. Entretanto, o tempo total de simulação obtido permanece elevado, devido a quantidade de operações necessárias para simular uma transformação quântica.

A implementação da biblioteca *qGM-Analyzer* em *C++* segue as otimizações introduzidas em [4], apenas alterando as estruturas de dados utilizados e fazendo uso de recursos nativos oferecidos pela linguagem visando a otimização da execução. Visando a compatibilidade da biblioteca com o ambiente *VPE-qGM*, é considerada a biblioteca *Boost 1.49.0* [1] para integração das duas linguagens envolvidas. Para a implementação do paralelismo foi utilizada A biblioteca *OpenMP* [3] implementa o paralelismo.

O módulo *Boost-Python* auxiliou no desenvolvimento da biblioteca qGM_C*-Analyzer* permitindo que sejam reutilizados no código *C++* tipos do *Python*, como as listas de valores utilizadas no ambiente *VPE-qGM*, e ainda realizando a conversão destes tipos para tipos da linguagem *C++*. Outra funcionalidade utilizada do módulo *Boost-Python* foi a geração da biblioteca compartilhada importada pelo ambiente em *Python*.

A implementação em *C++* manteve o mesmo algoritmo desenvolvido em [4], porém a forma de armazenamento dos dados e os algoritmos para acesso a esses dados foram modificados para diminuir a complexidade da biblioteca. Dentre os fatores que contribuíram para um melhor desempenho, destacam-se a utilização de vetores como alternativa as listas do *Python* e o fato de a linguagem *C++* ser compilada, permitindo ao compilador otimizações no código gerado.

Na implementação paralela, cada *thread* possui uma cópia privada da pilha, e as memórias de escrita e leitura são compartilhadas entre todos os *threads*, pois cada valor calculado é escrito em uma posição diferente da memória. A divisão dos *threads* é feita de forma a manter juntos os valores das matrizes necessários para o cálculo de uma posição. Para tal, iterações são divididas levando em conta o número de colunas da primeira matriz envolvida na transformação. Nos casos em que a transformação possui apenas uma matriz, os *threads* são divididos de forma diferente, para que o trabalho seja distribuído de forma balanceada. A definição do número de *threads* utilizadas pela biblioteca é definida por uma variável de ambiente (*OMP_NUM_THREADS*), gerando uma implementação mais flexível. Ou seja, dependendo da transformação quântica, tem-se um controle da granulosidade visando melhor desempenho da biblioteca.

Para validação e análise de desempenho da implementação da *qGM-Analyzer* em *C++*, foram desenvolvidos estudos de caso envolvendo sincronizações arbitrárias de transformações quânticas, contemplando sistemas entre 13 e 20 qubits.

A metodologia dos testes utilizada contempla, para cada estudo de caso, a realização de 15 simulações. A máquina utilizada na simulação possui as seguintes características: processador *Intel Core i7-3770 @ 3.4 GHz*, *8GB RAM* e sistema operacional *Ubuntu 12.04 64 bits*.

A principal comparação de desempenho se dá com a execução da biblioteca em diferentes números de *threads*. Os testes foram realizados com transformações *Hadamards*, representada por uma matriz densa de ordem 2^n, onde n é o número de *qubits* e transformações *Pauli X*, uma matriz esparsa de ordem 2^n. Estas execuções compreenderam duas etapas: (i) Geração dos valores não nulos associados ao correspondente vetor componente da matriz de definição da transformação quântica modelada; (ii) Multiplicação desses valores pelas amplitudes obtidas da estrutura de memória que modela o espaço de estados do sistema quântico. Nas simulações das *Hadamards*, devido ao elevado número de operações, pode-se facilmente observar um desempenho elevado, ou seja, o aumento no número de *qubits* diretamente relacionado com a diminuição significativa do tempo de execução. Nas transformações controladas *Pauli X* houve um aumento no tempo de execução. Justifica-se este fato pelo baixo número de operações en-

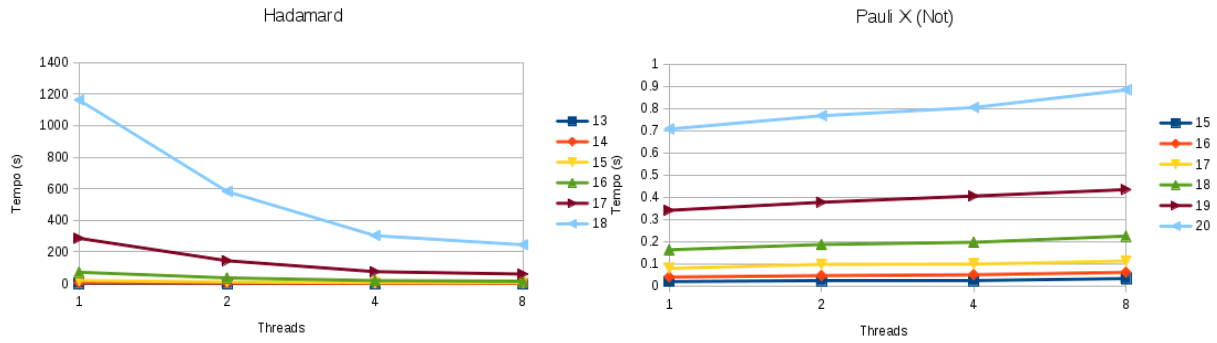

Figura 1: Tempos das transformações Hadamard e Pauli X em relação ao número de threads.

volvidas. Sendo a *Pauli X* uma matriz esparsa, seus valores zerados não são computados pelo algoritmo otimizado, tornando o *overhead* da criação e troca de contexto dos *threads* o principal custo da transformação, como observado no decremento de desempenho da Figura da *Pauli X*.

Os tempos de simulação, em cada estudo de caso, abrangendo os casos para 1, 2, 4 e 8 *threads* são descritos na Figura 1. Como analisado, os tempos de execução para as transformações *Hadamards* obtiveram os melhores desempenhos. As transformações Pauli X tiveram o seu tempo de execução aumentando conforme aumentava o número de *threads*, pois envolvem poucos cálculos, e seu tempo é dominado pela criação e troca de contexto dos *threads*.

As otimizações realizadas e a implementação paralela representaram um ganho significativo no tempo de execução das transformações quânticas, permitindo a simulação de transformações de 18 *qubits* com elevado número de operações, como no caso das Hadamards.

A continuidade do trabalho consiste na integração da biblioteca *qGM$_C$-Analyzer* com o ambiente *VPE-qGM* além da expansão para suporte a transformações *multiqubits* e na implementação de aplicações em outras bibliotecas disponíveis, para comparação de desempenho entre diferentes ferramentas para suporte a simulação quântica.

Palavras-chave: *VPE-qGM, Computação Quântica, OpenMP, Simulação Quântica*

Referências

[1] Boost, Boost 1.49.0 library documentation, *www.boost.org/doc/libs/1_49_0/*, (2012).

[2] A. Ávila; A. Maron; R. Reiser and M. Pilla, Extending the VirD-GM environment for the distributed execution of quantum processes, *Proc. of the XIII WSCAD-WIC*, (2012), 1-4.

[3] E. Ayguade and B. Chapman, "Introduction: Special Issue: OpenMP", Scientific Programming, 2003.

[4] A. Maron; A. Ávila; R. Reiser and M. Pilla, Introduzindo uma nova abordagem para simulação quântica com baixa complexidade espacial, *Anais do DINCON 2011*, (2011), 1-6.

[5] A. Maron; R. Reiser and M. Pilla, High-performance quantum computing simulation for the quantum geometric machine model, *Proceedings of CCGRID 2013*, (2013), 1-8.

[6] A. Maron; R. Reiser and M. Pilla, Quantum Processes: A Novel Optimization for Quantum Simulation, *TEMA Tendências em Matemática Aplicada e Computacional]*, 2013 (a ser publicado).

[7] M. Nielsen and I. Chuang, "Quantum Computation and Quantum Information", Cambridge University Press, 2000.

[8] R. Reiser and R. Amaral, The quantum states space in the qGM model, *Proc. of the III WECIQ*, (2010) 92-101.

[9] M. Schmalfuss; A. Maron; R. Reiser and M. Pilla, qGM$_C$-Analyzer: biblioteca para suporte à simulação quântica em C++, *Proc. of the XIII WSCAD-WIC*, (2012).

[10] W. Steeb and Y. Hardy, Problems and solutions in quantum computing and quantum information, *World Scientific*, (2004).

