Proceeding Series of the Brazilian Society of Applied and Computational Mathematics, Vol. 3, N. 1, 2015.
Trabalho apresentado no XXXV CNMAC, Natal-RN, 2014.

Simulação Quântica em Arquiteturas Multicore via Biblioteca qGM$_C$-Analyzer

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

RESUMO

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 [5].

O ambiente *VPE-qGM*, fundamentado no modelo de processos *qGM* (*Quantum Geometric Machine Model*) [4], 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 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.

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* [1]. 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 [1], apenas alterando as estruturas de dados utilizados e fazendo uso de recursos nativos oferecidos pela linguagem visando a otimização da execução. Para a implementação do paralelismo foi utilizada a biblioteca *OpenMP* [3].

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. As 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

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

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, foram desenvolvidos estudos de caso de transformações quânticas *Hadamard* e *Pauli X* [2], contemplando sistemas entre 13 e 20 qubits.

Para cada estudo de caso foram realizadas 15 simulações. A máquina utilizada nas simulações possui as seguintes características principais: processador *Intel Core i7-3770 @ 3.4 GHz* com *Hyper-Threading* abilitado e *Turbo Boost* desabilitado, *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. Podemos observar o *speedup* das simulações em relação a um *thread* na

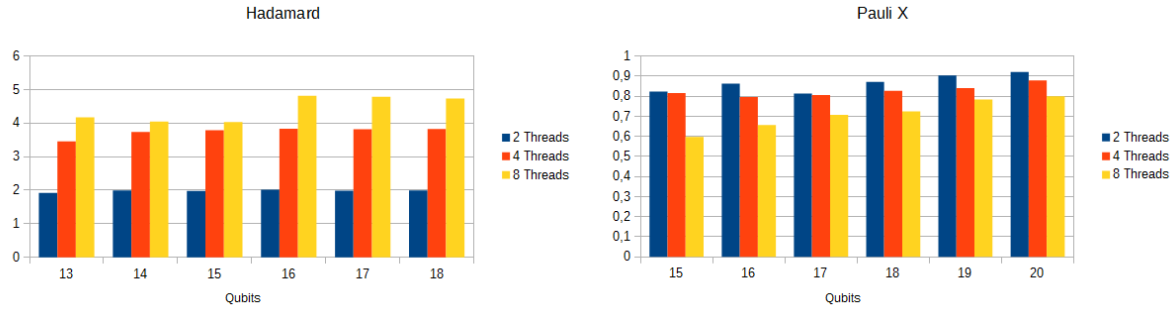

Figura 1: Speedup das transformações Hadamard e Pauli X em relação a um thread.

Figura 1. Nas simulações das *Hadamards*, observa-se que para as configurações de 2 e 4 *threads*, o *speedup* aproxima-se do ideal conforme aumenta-se o número de *qubits*. Para 8 *threads* o desempenho não segue o mesmo padrão, justificado pelo uso do *hyper-threading*. Nas transformações controladas *Pauli X* o *speedup* diminui conforme aumenta-se o número de *threads*. Justifica-se este fato pelo baixo número de operações envolvidas. 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.

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.

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

Referências

[1] *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.*

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

[3] *Chapman, B.; Jost, G. and Van Der Pas, R., "Using OpenMP: Portable Shared Memory Parallel Programming", The MIT Press, 2008.*

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

[5] *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).*

