Otimização de Simulação de Computação Quântica Através da Redução e Decomposição Baseados no Operador Identidade

Anderson Avila[1], **Renata H. S. Reiser**[1] **e Maurício L. Pilla** [1]

[1]PPGC – CDTec – UFPel
Pelotas – Brasil

{abdavila, reiser, pilla}@inf.ufpel.edu.br

Abstract. *A main obstacle in simulation of quantum algorithms is the exponential increase in the temporal and spatial complexities, especially in dense quantum transformations such as the Hadamard operator. In this work, new optimizations for the execution of quantum transformations in the Distributed Geometric Machine (D-GM) environment.Instead of executing them in a single step, they are decomposed and only values different from Identity operator are stored. As a benchmark, Hadamard Transformations were simulated up to 28 qubits in a GPU. When compared to our previous implementation, our new approach is* $10,829\times$ *faster and allows for the simulation of more qubits.*

Resumo. *Um dos maiores obstáculos para a simulação de algoritmos quânticos é o crescimento exponencial nas complexidades espaciais e temporais, especialmente em transformações quânticas densas como o operador Hadamard. Neste trabalho, são introduzidas novas otimizações para a execução de transformações quânticas no ambiente* Distributed Geometric Machine *(DGM).Ao invés de executá-las em um único passo, estas são decompostas e apenas os valores que diferem do operador Identidade são armazenados. Como benchmark, transformações Hadamard foram simuladas com até 28 qubits em uma GPU. Comparando à implementação anterior, os resultados foram* $10.829\times$ *mais rápidos e permitiram a simulação de um número maior de qubits.*

1. Introdução

A Computação Quântica (CQ) é um novo paradigma com o potencial de explorar fenômenos da mecânica quântica para prover enorme paralelismo. No entanto, computadores quânticos ainda encontram-se no ínicio de seu desenvolvimento, trabalhando com poucos bits quânticos, ou qubits, e indisponíveis para a grande maioria dos pesquisadores.

Enquanto os computadores quânticos não são uma realidade prática, o desenvolvimento e teste de novos algoritmos para estes sistemas pode ser feito através de processos analíticos ou simulação. Embora simulação iterativa possua diversas vantagens sobre processos analíticos, como facilidade de uso, a simulação de computação quântica sobre computadores clássicos é uma tarefa com altas complexidades espacial e temporal. Qubits são representados como vetores e transformações quânticas como matrizes, resultantes do produto tensor entre seus operadores. Assim, o tamanho das matrizes aumenta drasticamente com o número de qubits.

Simuladores vêm explorando o potencial de arquiteturas massivamente paralelas encontradas em GPUs [Gutierrez et al. 2010]. A simples execução de operações como multiplicações de matrizes já apresenta grandes *speedups* quando comparada ao uso de processadores de propósito geral, mas os requerimentos de memória torna-se um impecilho à escalabilidade.

Neste trabalho, a representação de transformações quânticas são melhoradas pelo uso inteligente do operador Identidade e pela divisão de operações independentes. Embora o número de passos necessários seja maior, o tempo de simulação é drasticamente reduzido. Os passos resultantes podem ser executados em paralelo na mesma GPU ou em sistemas diferentes, mesmo distribuídos.

Estas duas abordagens para melhorar o desempenho de simulações de algoritmos quânticos foram implementadas na *Distributed Geometric Machine* (D-GM) e avaliados experimentalmente com a transformada quântica de Hadamard, entre 21 e 28 qubits, com diferentes tamanhos de sub-passos. Esta transformada foi escolhida por apresentar o maior custo em termos de simulação, portanto, qualquer otimização que apresente bons resultados com ela irá beneficiar as simulações em geral. O melhor *speedup* relativo em relação à solução anterior foi obtido com 22 qubits e apresentou desempenho $10.829\times$ melhor.

Este artigo é organizado como segue. Primeiro, o ambiente D-GM e seus componentes são descritos na Seção 2. Após, fundamentos básicos de computação quântica são discutidos na Seção 3. A Seção 4 apresenta as otimizações introduzidas neste trabalho. O conceito de processos parciais mistos é introduzido na Seção 5. A implementação das otimizações na D-GM é detalhada na Seção 6. Resultados são analisados na Seção 7. Os trabalhos relacionados são descritos na Seção 8. Finalmente, as conclusões e trabalhos futuros são discutidos na Seção 9.

2. O Ambiente D-GM

O projeto D-GM propõe um *framework* completo para simulação de algoritmos quânticos, com interfaces gráficas para modelagem e um sistema de execução para simular sequencialmente ou em paralelo, usando GPUs e CPUs. A Figura 1 apresenta a organização do D-GM em seus diferentes níveis: *(i) Nível de Circuito Quântico:* usado na descrição de aplicações como circuitos quânticos; *(ii) Nível qGM:* engloba o VPE-qGM (Visual Programming Environment for the qGM Model), usado para descrever as computações no modelo *quantum Geometric Machine* (qGM); e *(iii) Nível D-GM:* implementa o gerenciamento de simulações distribuídas que se encarrega pela comunicação, escalonamento e sincronização das simulações.

3. Fundamentos de Computação Quântica

Algoritmos quânticos são modelados de acordo com conceitos matemáticos [Nielsen and Chuang 2003]. Um qubit é a unidade básica de informação e o mais simples sistema quântico. Um qubit é definido como um vetor de estado unitário e bidimensional, descritos na notação de Dirac [Nielsen and Chuang 2003]:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle. \quad (1)$$

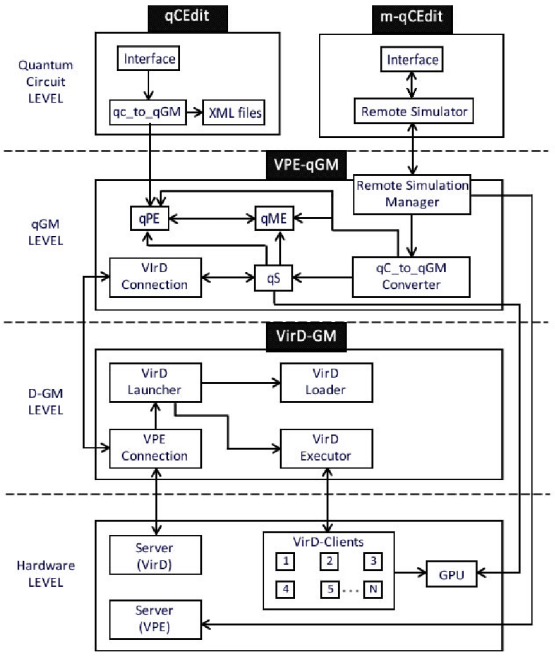

Figura 1. Estrutura do *Framework* de Simulação D-GM. [Avila et al. 2014b].

Os coeficientes α e β são números complexos para as amplitudes dos estados correspondentes na base computacional (espaço de estados), respeitando a condição $|\alpha|^2 + |\beta|^2 = 1$.

O espaço de estados de um sistema quântico com múltiplos qubits é obtido pelo produto tensor dos espaços de estados de seus subsistemas. Por exemplo, para um sistema quântico de dois qubits, $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ e $|\varphi\rangle = \gamma|0\rangle + \delta|1\rangle$, o espaço de estados corresponde ao produto tensor $|\psi\rangle \otimes |\varphi\rangle$, descrito por:

$$\alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle. \tag{2}$$

Transições de estados em um sistema quântico N-dimensional de transformações quânticas unitárias são associadas a matrizes quadradas de ordem N com 2^N elementos. A notação matricial dos operadores Identidade, Hadamard e Pauly X são, respectivamente:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \; I = \begin{pmatrix} 1 & 0 \\ 0 & \mathbf{1} \end{pmatrix} \; and \; X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \tag{3}$$

A aplicação de Hadamard a um estado quântico $|\psi\rangle$, denotado por $H|\psi\rangle$, gera um novo estado global:

$$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \times \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} \alpha + \beta \\ \beta - \beta \end{pmatrix}. \tag{4}$$

Transformações quânticas aplicadas a diferentes qubits simultaneamente tem sua matriz definida pelo uso do produto tensor entre os operadores.

Pela composição e sincronização de transformações quânticas, é possível executar computações que exploram o potencial do paralelismo quântico. No entanto, o cres-

cimento exponencial na memória implica em limites na simulação de sistemas quânticos multidimensionais. Desta forma, simulações usando diretamente a notação matricial, sem otimizações, aplicam-se apenas a um pequeno número de qubits.

4. Redução da Complexidade Espacial e Temporal

Em trabalhos anteriores [Maron et al. 2012, Maron et al. 2013, Avila et al. 2014b], processos relacionados a TQs n-dimensionais eram definidos por um conjunto de matrizes básicas de baixa ordem para reduzir a memória usada nas simulações devido ao crescimento exponencial da sua representação por uma única matriz ($2^n \times 2^n$). Elementos da TQ ncessários para sua computação eram gerados em tempo de execução por iterações sobre estas matrizes, simulando o comportamento do produto tensor. No entanto, o tempo total computacional gasto nestas iterações é alto quando se dá a execução de TQs compostas por muitos operadores densos.

As otimizações propostas neste artigo estão principalmente relacionadas à redução da complexidade espacial e temporal associada a simulação de TQ pelo uso inteligente do operador Identidade. Abordagens distintas foram usadas para alcançar bons resultados, e elas são descritas nas sub-seções a seguir.

4.1. Otimização de TQs baseada no operador Identidade

A primeira otimização explora o comportamento do produto tensor entre o operador Identidade (I) e outros operadores. Neste casos, o operador Identidade não só replica os dados dos outros operadores como também introduz esparcialidade na TQ. Portanto é possível armazenar somente a expansão do produto tensor entre operadores diferentes de I gerando uma matriz reduzida (MR), dimuindo então a complexidade espacial das TQs com estas características. Um exemplo do comportamento deste operador é mostrado na Eq. (5) para a transformação $I \otimes H$.

$$I \otimes H = \begin{pmatrix} 1 & 0 \\ 0 & \mathbf{1} \end{pmatrix} \otimes \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & \mathbf{1} & \mathbf{1} \\ 0 & 0 & \mathbf{1} & \mathbf{-1} \end{pmatrix} \tag{5}$$

Uma vez que a ordem da MR é menor que a dimensão do estado, não é possível realizar a multiplicação entre matriz/vetor como normalmente é feito para calcular as novas amplitudes. Portanto, para usar esta otimização é preciso adotar uma abordagem diferente. As informações sobre o cálculo de uma nova amplitude são descritas à seguir:

- Cada bit da representação binária da posição da nova amplitude está relacionado a um operador da TQ. O mais significativo ao operador do primeiro qubit, segundo mais significativo ao operador do segundo qubit, e assim por diante;
- Bits relacionados a operadores diferente de I são bits-ativos;
- A linha da MR usada para o cálculo é determinada pela concatenação dos bits-ativos;
- Cada elemento desta linha é multiplicado por uma amplitude do estado de leitura, determinadas substituindo os bits que representam a coluna do elemento pelo bits-ativos da nova amplitude.

- O valor da nova amplitude é determinado pela soma destas multiplicações.

Por exemplo, considere a transformação $H \otimes I \otimes H$, a qual tem operadores diferentes de I no primeiro e no terceiro qubit. A sua MR é definida por $H \otimes H$ e o cálculo da amplitude da posição 6 do novo estado é demonstrado a seguir.

Nova Amplitude: 6 (**110**) $\rightarrow$ Linha da MR: 2 (**10**);

Na expressão descrita abaixo, EI e NE indicam o estado inicial e o novo estado, respectivamente. Para melhor vizualização, foram usados indices coloridos na forma binários.

$$\begin{aligned} \mathbf{NE}[110] \quad = \quad & MR[10][00] \times EI[010] + MR[10][01] \times EI[011] + \\ & MR[10][10] \times EI[110] + MR[10][11] \times EI[111] \end{aligned}$$

Embora este conceito otimize a representação de TQs envolvendo operadores I, nem todas TQs apresentam operadores I ou uma quantidade suficiente que torne possível representá-la através de uma única matriz na memória. E para superar esta limitação, a próxima otimização considera a decomposição de TQs.

4.2. Decomposição de TQs

Uma TQ n-dimensional pode ser decomposta aumentando o número de passos para o cálculo da mesma, permitindo controlar a quantidade de operadores I em cada passo, preservando o comportamento e propriedades da TQ.

Na figura 2 é mostrado que a transformação $H \otimes H$ pode ser decomposta em dois passos, $H \otimes I$ e $I \otimes H$, mantendo o mesmo comportamento independente da ordem de composição destes passos.

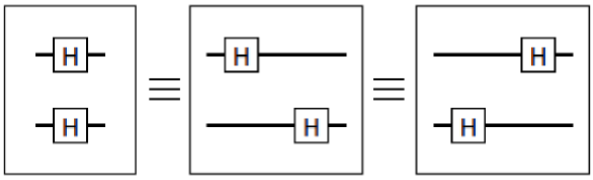

Figura 2. Transformação não-controlada decomposta

TQs controladas também podem ser decompostas desde que os operadores conservem os controles associados a eles, como mostrado na figura 3.

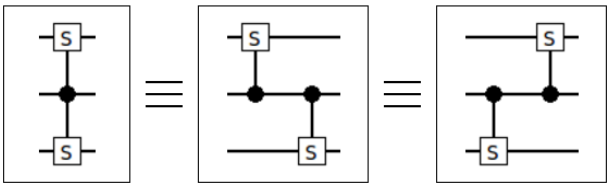

Figura 3. Transformação controlada decomposta

Usando estes conceitos para otimizar a representação de TQs, é possível criar um algoritmo eficiente que não precise simular o comportamento do produto tensor. Pois a

complexidade espacial pode ser reduzida limitando o número de operadores diferentes de I presentes em cada passo da decomposição, tornando possível a respresentação de cada passo por uma única matriz.

5. Processos Mistos Parciais

Apessar de ser possível modelar TQs com baixa complexidade espacial, usando as abordagens mostradas na seção anterior, o tamanho dos estados de leitura/escrita pode se tornar um fator limitante para simulação de TQs n-dimensionais, pois também crescem de forma exponecial (2^n). Por exemplo, uma TQ 28-dimensional precisa 4 GiB de memória para armazenar ambos estados.

Uma vez que a memória das GPUs normalmente são menores que a memória RAM principal, é necessário adotar uma abordagem que forneça escalabilidade para a simulação de TQs multi-qubits;

O conceito de Processos Mistos Parcias (MPP), apresentando em [Avila et al. 2014a] e [Avila et al. 2015], provê controle sobre o tamanho dos estados de leitura/escrita no cálculo de uma TQ, contribuindo para o aumento da escalabilidade. Baseando-se neste conceito, TQs com mais qubits que o limite suportado pela memória da GPU, podem ter seus estados particionados em 2^p sub-estados, onde p indica o número de qubits acima do limite, tornando possível a simualação da TQ.

Usando as otimizações descritas anteriormente, o número de sub-estados de leituras que cada sub-estado de escrita precisa acessar para realizar o cálculo de suas amplitudes é 2^r, sendo r o número de operadores afetados pelo particionamento. Por operadores afetados entende-se o número de operadores diferentes de I presentes nos p primeiros qubits do passo sendo computado. Logo, passos que não possuam operadores afetados precisam somente do correspondente sub-estado de leitura, o que os torna totalmente independentes.

Para manter a consistência do resultado, cada passo afetado precisa calcular o resultado de todos os sub-estados antes de prosseguir para o próximo passo devido as dependências existentes entre eles. E os passos não-afetados podem ser calculados da mesma forma ou podem ter cada sub-estado calculado de forma iterativa pois não há dependências, ou seja, cada sub-estado de escrita precisa somente do correspondente sub-estado de leitura para realização do cálculo.

No caso de controles serem afetados, somente os sub-estados que satisfaçam estes controles precisam ser calculados e acessados para leitura.

6. Implementações

A abordagem considerando a redução e decomposição de TQs baseadas no operador Identidade visa a redução da complexidade espacial e temporal na simulações de aplicações quântica muli-qubits.

A estratégia adotada para reduzir a complexidade espacial é baseada nas abordagens mostradas na seção 4, e pode ser dividida em duas partes:

(1) classificação da TQ em grupos, dividindo operadores não-controlados e com controles distintos.

(2) passos da TQ são formados por operadores que pertençam ao mesmo grupo e atuem em qubits consecutivos respeitando o limite de operadores por passo estabelecido. Operadores afetados e não-afetados não podem fazer parte do mesmo passo, caso a memória tenha sido particionada.

Um exemplo é mostrado na Figura 4 para uma TQ de 9 qubits considerando limites de 3 operadores e de 8 qubits para execução. A TQ primeiro é dividida em 3 grupos e então em 5 passos. O grupo 1 gera dois passos apesar de ter 3 operadores consecutivos, pois o primeiro qubit é afetado pelo particionamento da memória.

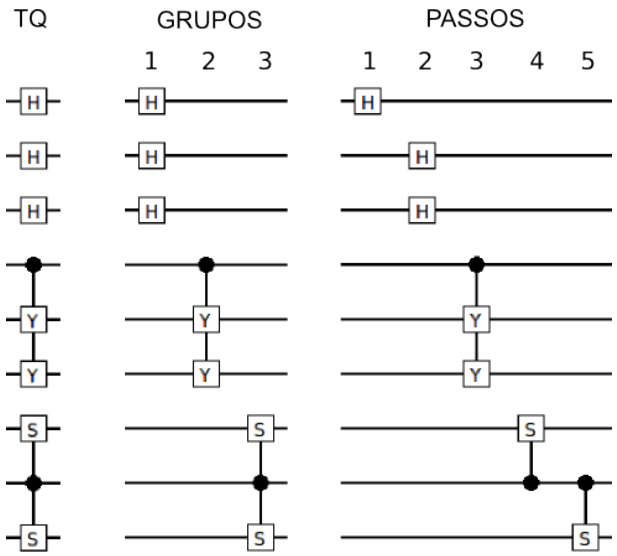

Figura 4. Exemplo de decomposição para uma TQ.

Depois da decomposição da TQ de acordo com os passos descritos acima, o cálculo dos passos é realizado.

Como descrito na Seção 5: passos afetados são calculados um por um, sendo realizada uma chamada de kernel para cada combinação entre sub-estados de escritas e sub-estados de leituta necessários para o cálculo. Todos os passos não-afetados são calculados um sub-estado por vez, realizando chamandas de kernel de forma iterativa a fim de reduzir a comunicação entre host e GPU, uma vez que o sub-estado de escrita resultante do cálculo de um passo pode ser mantido na memória da GPU e servir como sub-estado de leitura para o próximo passo.

6.1. Kernel CUDA

Cada chamada do kernel CUDA recebe os seguintes parâmetros para execução:

- Estados (sub-estados) de leituta/escrita;
- Matriz reduzida do passo a ser computado;
- Valores e posições dos controles (se existirem);
- Informações de acesso as estruturas que possuem os dados citados acima.

A computação do kernel CUDA é dividida em 5 passos, descritos a seguir.

Passo 1: Identificação do $lineId$ de cada *thread* (Figura 5), calculado usando informação sobre a *thread* atual e do controle. O $lineId$ define qual amplitude será calculada por cada *thread*.

```
long read_shift = arg[SHIFT_READ];
long shift_write = arg[SHIFT_WRITE];
long lineId = (blockIdx.y * gridDim.x + blockIdx.x) *
              blockDim.x + threadIdx.x;

if (arg[CTRL_COUNT]){
   for (i = arg[CTRL_COUNT] - 1; i >= 0 ; i--){
      lineId = (lineId*2) - (lineId &
               (1 << (ctrl_pos[i]) - 1));
   }
   lineId = lineId | arg[CTRL_VALUE];
}
lineId = lineId | shift_write;
```

Figura 5. Passo 1: definição dos $lineId$ para cada *thread*.

Passo 2: Inicialização de variáveis locais a cada *thread*(Figura 6).

```
long p = arg[MAT_START];
long size = arg[MAT_SIZE];
long shift = arg[SHIFT];
long read_mask = (size - 1) << shift;
long inc = 1 << shift;
long read_pos = (lineId & ~read_mask) + (p << shift);
long base = ((lineId & read_mask) >> shift) * size;
long end = arg[MAT_END];
long read_shift = arg[SHIFT_READ];
```

Figura 6. Passo 2: inicialização de variáveis locais.

Passo 3: Computação da nova amplitude parcial usando as variáveis do passo anterior, que definem acessos à matriz e estados (Figura 7).

```
cuFloatComplex accum = make_cuFloatComplex(0.0,0.0);
for(; p < end; p++){
   accum= cuCaddf(accum,
          cuCmulf(readMem[read_pos - read_shift],
          matrix[base+p]));
   read_pos += inc;
}
```

Figura 7. Passo 3: cálculo da nova amplitude.

Passo 4: Armazenamento e acumulação das novas amplitdues, calculadas e escritas na memória global da GPU (Figura 8).

```
lineId -= shift_write;
if (arg[ACUMM])
   writeMem[lineId]= cuCaddf(writeMem[lineId],accum);
else
   writeMem[lineId]= accum;
```

Figura 8. Passo 4: armazenamento e acumulação da nova amplitude.

Passo 5: Cópia das amplitudes das posições complementares aos controles do estado de leitura para o estado de escrita (Figura 9).

```
if(arg[CTRL_CMPL]){
   lineId = lineId & (~arg[CTRL_MASK]);
   for (i = 0; i < arg[CTRL_CMPL]; i++){
      p = lineId | ctrl_cmpl[i];
      writeMem[p] = readMem[p];
   }
}
```

Figura 9. Passo 5: cópia das amplitudes complementares ao controle.

7. Resultados

A principal contribuição deste trabalho é validada pela simulação de transformações Hadamard de 21 até 28 qubits, um operador em cada qubit. Considerando parâmetros de limite de operadores por passo variando de 1 até 5 e limite de qubits para simulação em GPU de 26 até 28 qubits com o intuito de analisar o comportamento do novo algoritmo do ambiente D-GM. A escolha da transformação Hadamard se dá pelo fato dela ser a operação com maior custo computacional na simulação de algoritmos quânticos.

Os tempos médios de execução, para todas as combinações de transformações e parâmetros, foram obtidos depois de 40 execuções, descartando os 5 menores e maiores tempos obtidos. Os teste foram realizados em um desktop com processador Intel Core i7-3770, 8 GiB RAM, GPU NVIDIA TeslaK20 e sistema operacional Ubuntu Linux 12.04, 64 bits, com CUDA 5.0.

7.1. Tempo de Execução

Os tempos de simulação obtidos, sem considerar limitação de qubits para execução, são mostrados na Figuras 10 (21 até 24) e 11 (25 até 28), salienta-se que escalas diferentes foram utilizadas para melhor visualização. Observa-se que o tempo para qualquer transformação Hadamard diminui quando o limite de operadores varia de 1 até 3, e aumenta quando varia de 3 até 5. A tendência é continuar crescendo conforme aumenta-se este limite, mostrando que simulações com limite de operadores igual a 3 obtêm o melhor desempenho neste hardware.

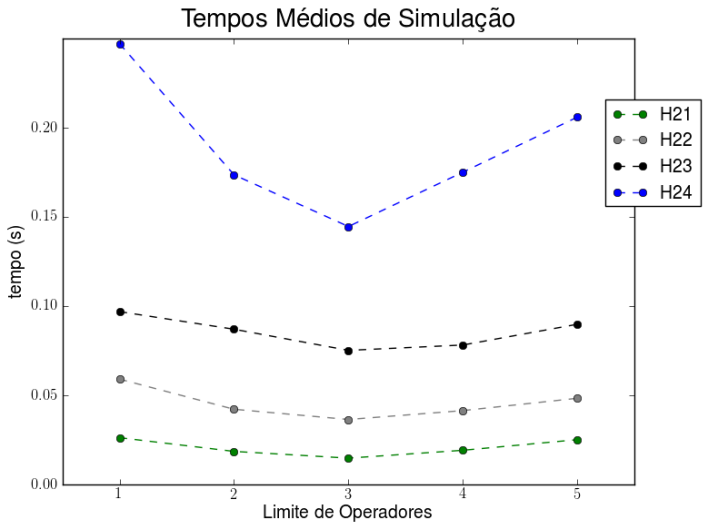

Figura 10. Hadamard 21-24

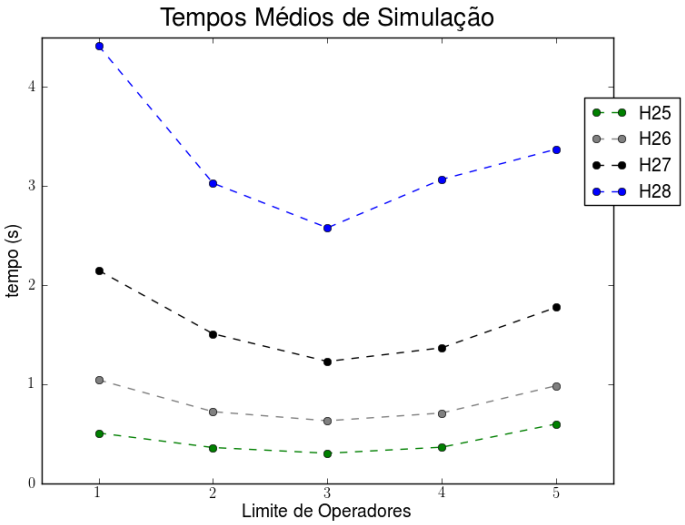

Figura 11. Hadamard 25-28

A Figura 12 mostra os tempos de simulação para as transformações Hadamards de 27 e 28 qubits, considerando limites de qubits para execução de 26 até 28 qubits. Como esperado as implementações usando MPPs permitem a simulação mesmo quando o limite de qubits para execução é menor que o número de qubits da transformação sendo calculada. No entanto, o tempo de execução aumenta conforme a quantidade de qubits passados do limite, pois haverá mais operadores afetados pelo particionamento do estado.

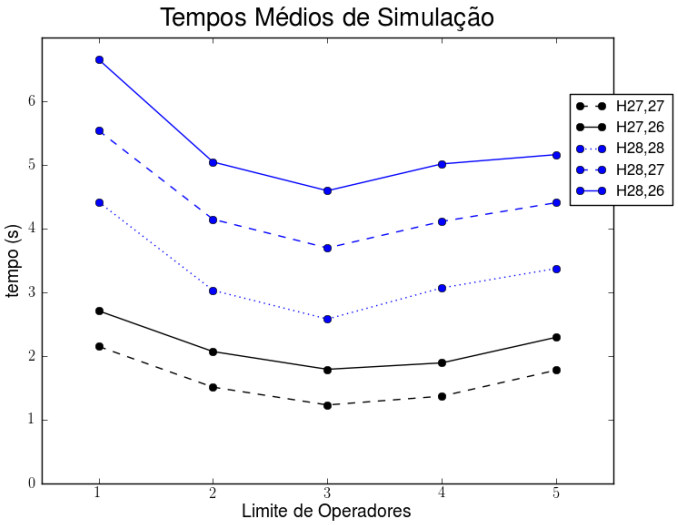

Figura 12. Hadamard 25-28, Limite 27-28

Tempos médios de simulação usando nosso método anterior foram medidos para transformações Hadamard de 21 e 22 qubits com tempos de 110,407 s e 395,951 s respectivamente. O speedup relativo obtido comparando nosso melhor resultado neste trabalho com o método anterior foi de $\approx 10,829\times$. Este speedup tende a escalar com o número de qubits, pois a taxa de crescimento do tempo conforme o aumento do número de qubits é de $\approx 2\times$, enquanto no método anterior é de $\approx 3.6\times$.

8. Trabalhos Relacionados

Alguns simuladores paralelos já foram propostos para acelerar a simualção de algoritmos quânticos. Sejam eles baseados em clusters ou GPUs, bons resultados podem ser obtidos.

8.1. Simulação usando super-computadores

O MPQCS (*Massive Parallel Quantum Computer Simulator*) [Raedt et al. 2006] é um simulador de computação quântica em MPI (*Message Passing Interface*), voltado para *clusters* ou redes de estações. Algoritmos são descritos através de um conjunto de transformações quânticas universais tais como $\{H, S, T, CNOT\}$. Através de combinações destas transformações, torna-se possível descrever qualquer algoritmo quântico.

A simplicidade do uso de um conjunto restrito de transformações facilita a otimização agressiva do simulador. No entanto, o uso de um conjunto universal de transformações torna difícil desenvolver algoritmos mais complexos.

Em 2010, o MPQCS foi capaz de simular o algoritmo de Shor com 42 qubits no supercomputador JUGENE [Henkel 2010], fatorando o número 15707 em 113×139. Para tanto, foram usados 262.144 processadores. O tempo de execução e memória empregada não foram divulgados.

8.2. Simulação usando a linguagem de programação CUDA

O simulador descrito em [Gutierrez et al. 2010] utiliza o *framework* CUDA para explorar a natureza paralela de algoritmos quânticos. Neste trabalho, as computações relacionadas à evolução do sistema quântico são executadas por milhares de *threads* dentro de uma GPU.

Para tanto, um conjunto de transformações de 1 e 2 qubits é definido, sendo uma solução mais genérica em termos de suporte a transformações quando comparado à proposta descrita em [Raedt et al. 2006]. O uso de um conjunto mais expressivo de transformações quânticas expande as possibilidades de descrição de algoritmos quânticos.

A maior limitação da simulação de algoritmos quânticos em GPUs está relacionada à capacidade de memória. Gutierrez et al. [Gutierrez et al. 2010] simularam no máximo 26 qubits.

9. Conclusões e Trabalhos Futuros

Neste artigo, foi apresentada uma nova abordagem para reduzir a complexidade espacial e temporal na simulação de computação quântica. Pelo uso da otimização de operadores Identidade, decomposição de transformações quânticas e de processos mistos parciais, foi possível simular um grande número de qubits em uma única GPU.

Experimentos com transformações Hadamard mostraram que é possível realizar simulações de até 28 qubits em uma única GPU, tendo como fator limitante a memória RAM principal. Quando comparado com nosso trabalho anterior, o melhor *speedup* relativo foi obtido para a Hadamard de 22 qubits, $10.829 \times$ mais rápido que nossa abordagem anterior usando o mesmo *hardware*.

Em trabalhos futuros, pretende-se estender a abordagem desenvolvida neste trabalho para simulações em ambientes heterogêneos distribuídos compostos por vários

computadores com GPUs. Também será simulado o comportamento de agentes usando computação quântica fuzzy [Raghuvanshi and Perkowski 2010].

Referências

Avila, A., Schmalfuss, M., Reiser, R., and Pilla, M. (2014a). Distributed simulation of quantum algorithms via GPUs. In *Proc. of the XV Symposium on Computational Systems, (WSCAD-WIC)*, pages 1–12, São José dos Campos. IEEE CPS.

Avila, A., Schmalfuss, M., Reiser, R., Pilla, M., and Maron, A. (2015). Optimizing quantum simulation for heterogeneous computing: a hadamard transformation study. *Journal of Physics Conference Series*, pages 1–17. submitted.

Avila, A. d., Maron, A., Reiser, R. H. S., Pilla, M., and Yamin, A. (2014b). GPU-aware distributed quantum simulation. In *Symposium on Applied Computing*, pages 860–865, Gyeongju. Proc. of the 29th ACM Symposium on Applied Computing (SAC).

Gutierrez, E., Romero, S., Trenas, M., and Zapata, E. (2010). Quantum computer simulation using the cuda programming model. *Computer Physics Communications*, pages 283–300.

Henkel, M. (2010). Quantum computer simulation: New world record on jugene. Available at http://www.hpcwire.com/hpcwire/2010-06-28/ quantum _ computer_simulation_new_world_record_on_jugene.html (feb. 2013).

Maron, A., A., Reiser, R., Pilla, M., and Yamin, A. (2012). Quantum processes: A new interpretation for quantum transformations in the VPE-qGM environment. In *CLEI 2012*, pages 1–10. IEEE Computer Society - Conference Publishing Services.

Maron, A. K., Reiser, R. H. S., and Pilla, M. L. (2013). Correlations from conjugate and dual intuitionistic fuzzy triangular norms and conorms. In *CCGRID 2013 IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 1–8, NY. IEEE.

Nielsen, M. A. and Chuang, I. L. (2003). *Computação Quântica e Informação Quântica*. Bookman.

Raedt, K. D., Michielsen, K., Raedt, H. D., Trieu, B., Arnold, G., Richter, M., Lippert, T., Watanabe, H., and Ito, N. (2006). Massive parallel quantum computer simulator. http://arxiv.org/abs/quant-ph/0608239.

Raghuvanshi, A. and Perkowski, M. (2010). Fuzzy quantum circuits to model emotional behaviors of humanoid robots. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–8.