Velocidade de processamento do Algoritmo de Grover usando CUDA-Q

Frank Acasiete e Anderson Buarque

***Resumo*— Neste trabalho, vamos mostrar o algoritmo de Grover implementado com o pacote CUDA-Q, utilizando GPUs Nvidia para obter resultados mais rápidos do que ao executar este algoritmo com uma CPU, com diferentes quantidades de qubits.**

***Palavras-Chave*— Computação quântica, circuito quântico, GPU.**

***Abstract*— In this work, we will demonstrate the Grover's algorithm implemented with the CUDA-Q package, leveraging Nvidia GPUs to achieve faster results compared to executing this algorithm on a CPU, for varying numbers of qubits.**

***Keywords*— Quantum computing, quantum circuit, GPU.**

I. Introdução

Um marco crucial na implementação de computadores quânticos foi o conceito de computação reversível, introduzido por Bennett [1]. Essa ideia serviu como um modelo fundamental. Além disso, Toffoli [2] fez uma contribuição significativa ao desenvolver um conjunto universal de portas para esse modelo no contexto da computação clássica.

Posteriormente, o Algoritmo de Shor [3] gerou grande interesse na comunidade acadêmica. Ele demonstrou a capacidade de resolver eficientemente problemas considerados desafiadores para computadores clássicos, como o Problema da Fatoração de Inteiros e o Problema do Logaritmo Discreto, quando executado em um computador quântico.

No trabalho de Grover [4], foi demonstrado que um computador quântico pode encontraron um elemento em um banco de dados não ordenado usando apenas $O\sqrt{N}$ consultas. Essa é uma melhoria significativa em relação aos algoritmos clássicos, que exigiriam $O(N)$ consultas. Esse resultado teve um grande impacto na pesquisa em algoritmos quânticos e mostrou o potencial das máquinas quânticas para resolver problemas de busca de forma mais eficiente.

Atualmente, estamos na era NISQ (Noisy Intermediate-Scale Quantum) [5]. Isso significa que o uso de computadores quânticos é limitado pela presença de qubits ruidosos, o que impede a execução de experimentos quânticos de alta complexidade. Diante dessa limitação, a pesquisa busca ativamente alternativas para executar algoritmos quânticos. Uma das soluções mais promissoras é o uso de simuladores quânticos, como os oferecidos pela IBM e outras empresas. Esses simuladores são frequentemente construídos utilizando GPUs, que demonstram um desempenho excepcional no processamento de grandes volumes de dados.

Este trabalho está organizado da seguinte forma, na Seção II falaremos do algorito de Grover e do CUDA, na Seção III mostramos os resultados da implementação e na Seção IV damos uma breve conclusão do trabalho.

Frank Acasiete, QuIIN, Senai-Cimatec, Salvador-BA, e-mail: frank.quispe@fbter.org.br; Anderson Buarque, QuIIN, Senai-Cimatec, Salvador-BA, e-mail: anderson.correia@fieb.org.br. Este trabalho foi financiado total pelo projeto QIN-AFCCT-FC-CC-2024-6-11-1 apoiado pelo QuIIN-Inovação Industrial Quântica, Centro de Competência EMBRAPII CIMATEC em Tecnologias Quânticas, com recursos financeiros do edital MCTI número 053/2023, firmado com a EMBRAPII.

II. Preliminares

Nesta seção, discutiremos o Algoritmo de Grover e o CUDA, pois são as ferramentas essenciais que empregaremos para alcançar os resultados que serão mostrados neste trabalho.

A. Algoritmo de Grover

O algoritmo de Grover é uma ferramenta poderosa na computação quântica, especialmente para problemas de busca e otimização onde a eficiência é crucial. Ele consegue encontrar um elemento marcado com alta probabilidade utilizando apenas $O(\sqrt{N})$ avaliações da função, uma vantagem significativa em comparação com a busca clássica, que exigiria $O(N)$ avaliações.

No entanto, a principal barreira para que o algoritmo de Grover demonstre uma aceleração prática é que seu ganho quadrático é, na maioria das vezes, muito modesto para superar a grande sobrecarga dos computadores quânticos de curto prazo (os chamados dispositivos NISQ).

Apesar disso, em um experimento notável conduzido por [6], o algoritmo de busca de Grover foi implementado com sucesso em um sistema quântico com quatro estados. Esse feito representou um passo importante na demonstração prática da eficiência dos algoritmos quânticos em comparação com seus equivalentes clássicos.

No algoritmo de Grover, consideramos uma função [7]:

$$f : \{0, 1, \ldots, N-1\} \rightarrow \{0, 1\}$$

como entrada.

Imagine uma base de dados não estruturada. Nela, o domínio da função representa os índices que você pode consultar. A função $f(x)$ nos dirá se os dados apontados por um índice x específico satisfazem os critérios da sua busca. Para simplificar, vamos assumir que apenas um único índice, que chamaremos de ω, satisfaz a condição $f(x) = 1$. Nosso objetivo é, então, identificar esse índice ω.

Para interagir com essa função f, usamos uma sub-rotina especial, conhecida como oráculo, na forma de um operador

unitário U_ω. Este operador age de forma bem específica sobre um estado quântico $|x\rangle$:

$$\begin{cases} U_\omega |x\rangle = -|x\rangle & \text{para } x = \omega \text{ isto é, } f(x) = 1, \\ U_\omega |x\rangle = |x\rangle & \text{para } x \neq \omega \text{ isto é, } f(x) = 0. \end{cases} \tag{1}$$

Este comportamento pode ser representado de forma mais compacta. Utilizando o espaço de estados N-dimensional $\mathcal{H}$, que é fornecido por um registrador com $n = \lceil \log_2 N \rceil$ qubits, a ação do oráculo é frequentemente descrita como:

$$\boxed{U_\omega |x\rangle = (-1)^{f(x)} |x\rangle}. \tag{2}$$

Nesse algoritmo, utilizamos um espaço de estado N-dimensional $\mathcal{H}$, que é implementado por um registrador contendo $n = \lceil \log_2 N \rceil$ qubits. Dessa forma, o operador U_ω atua sobre os estados quânticos representados por esses qubits, aplicando um sinal negativo (-1) apenas ao estado $|x\rangle$ para o qual $f(x) = 1$. Se $f(x) = 0$, o estado $|x\rangle$ permanece inalterado.

O oráculo U_ω tem seu comportamento definido pela seguinte equação:

$$\boxed{U_\omega : |x\rangle |a\rangle = |x\rangle |a \oplus f(x)\rangle}, \tag{3}$$

onde $|x\rangle \in \mathcal{H}^N$, $|a\rangle \in \mathcal{H}^2$, e $\oplus$ representa a operação XOR.

O algoritmo de Grover é capaz de produzir o valor ω com uma probabilidade de, no mínimo, 1/2, utilizando apenas $O(\sqrt{N})$ aplicações do operador U_ω. A matriz de Grover, denotada por G, é definida por:

$$\boxed{G |x\rangle = \left(\frac{2}{N} - 1\right) |x\rangle + \frac{2}{N} \sum_{i \neq x} |i\rangle}. \tag{4}$$

O operador de evolução de Grover, U_G, é dado por

$$\boxed{U_G = G\, U_\omega}. \tag{5}$$

A condição inicial do algoritmo é

$$\boxed{|\psi_0\rangle = |G\rangle\ |-\rangle}, \tag{6}$$

onde $|-\rangle = (|0\rangle - |1\rangle)/\sqrt{2}$ e $|G\rangle = \frac{1}{\sqrt{N}} \sum_{u=0}^{N-1} |u\rangle$. O algoritmo de Grover instrui a aplicação iterativa do operador U_G por aproximadamente $\left\lfloor \frac{\pi}{4}\sqrt{N} \right\rfloor$ vezes.

Em seguida, medimos o primeiro registrador na base computacional. O resultado x terá uma probabilidade de, no mínimo, $1 - \frac{1}{N}$.

Para o caso de $m = 2$ elementos marcados, seja A o conjunto desses elementos marcados w_i. O operador oráculo, nesse cenário, é similar ao do caso de apenas um elemento marcado.

$$\begin{cases} U_\omega |x\rangle = -|x\rangle & \text{para } x = \omega \in A \text{ isto é, } f(x) = 1, \\ U_\omega |x\rangle = |x\rangle & \text{para } x \neq \omega \notin A \text{ isto é, } f(x) = 0. \end{cases} \tag{7}$$

Para este caso (referindo-se ao algoritmo de Grover com m elementos marcados), o tempo ótimo para encontrar um elemento marcado é

$$\boxed{t_{\text{opt}} = \left\lfloor \frac{\pi}{4}\sqrt{\frac{N}{m}} \right\rfloor}. \tag{8}$$

Ao final, medimos o primeiro registrador na base computacional e o resultado será um elemento em A com probabilidade maior ou igual a $1 - \frac{m}{N}$.

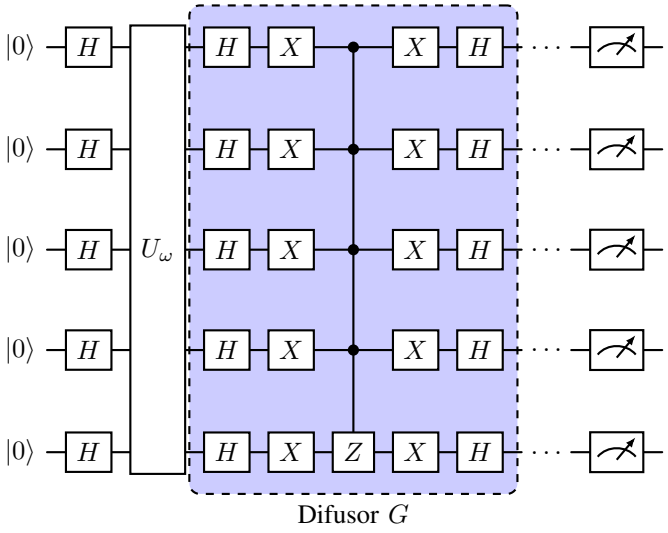

Fig. 1. Circuito del algoritmo de Grover.

Na Fig. 1 apresentamos o circuito do Algoritmo de Grover para o caso de 5 qubits, incluindo o operador difusor G.

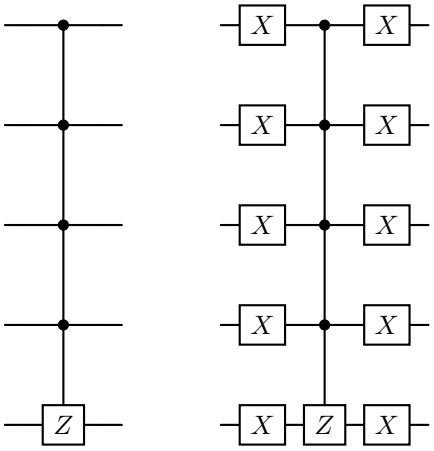

Fig. 2. Oráculos com elementos marcados $\omega_1 = 11111$ e $\omega_2 = 00000$.

Na Fig. 2, apresentamos o oráculo U_ω com os elementos marcados ω_1 e ω_2 que foram escolhidos.

B. CUDA

CUDA, que significa *Compute Unified Device Architecture*, é uma plataforma de computação paralela desenvolvida pela Nvidia. Ela engloba um compilador e um conjunto de ferramentas de desenvolvimento que permitem aos programadores utilizar uma variação da linguagem C (conhecida como CUDA C) para criar algoritmos executados diretamente nas GPUs da Nvidia [8].

O principal objetivo do CUDA é capitalizar as vantagens das GPUs sobre as CPUs de uso geral. Ele faz isso explorando o paralelismo massivo oferecido pelos múltiplos núcleos de uma

GPU, que permitem o lançamento e a execução simultânea de um número extremamente alto de threads.

Por isso, se uma aplicação for concebida com diversas threads executando tarefas independentes — um cenário que as GPUs naturalmente dominam ao processar gráficos —, essas unidades podem proporcionar um desempenho excepcional em uma vasta gama de áreas. Exemplos incluem desde a biologia computacional até a criptografia, demonstrando a versatilidade e o poder do paralelismo em GPU.

1) CUDA-Q: CUDA-Q é uma plataforma de desenvolvimento quântico de código aberto projetada para orquestrar o hardware e o software necessários para executar aplicações de computação quântica úteis em larga escala. Sua característica mais marcante é o modelo de programação híbrido, que permite a computação conjunta em recursos de GPU, CPU e QPU (Unidade de Processamento Quântico) a partir de um único programa quântico [9].

O CUDA-Q é notavelmente "agnóstico em qubits". Isso significa que ele se integra de forma transparente a todas as QPUs e modalidades de qubits existentes. Além disso, oferece simulações aceleradas por GPU quando o hardware quântico apropriado ainda não está disponível. A plataforma também estende as ferramentas de simulação muito além da era NISQ, pavimentando o caminho para a supercomputação quântica em larga escala e com correção de erros.

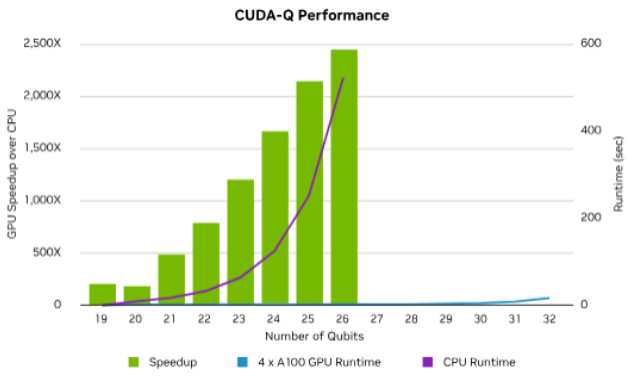

Fig. 3. Vantagem da GPU. Fonte: nvidia.com/cuda-q.

III. Resultados

Agora, apresentamos os resultados da implementação do CUDA-Q. Para isso, utilizamos o simulador quântico KUATOMU do Senai-Cimatec. Este simulador é capaz de simular até 35 qubits e é composto por um processador Intel 8260 de 24 núcleos e quatro GPUs Nvidia V100S, cada uma com 32GB. Para fins de comparação, também implementamos o algoritmo de Grover em CUDA-Q sem o uso das GPUs.

Iremos demonstrar dois casos distintos com diferentes números de qubits: um com apenas um elemento marcado e outro com dois elementos marcados. No primeiro caso, definiremos ω_1 como o estado em que todos os bits têm valor 1. Para o segundo caso, utilizaremos ω_2 com todos os bits tendo valor 0. Em ambos os cenários, empregamos 2000 shots para obter os resultados. As Figs. 4 e 5 ilustram os resultados da simulação,

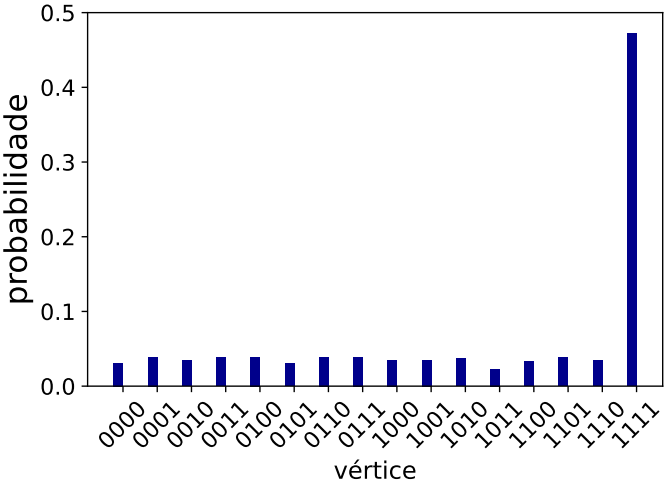

Fig. 4. Resultados da simulação com um elemento marcado.

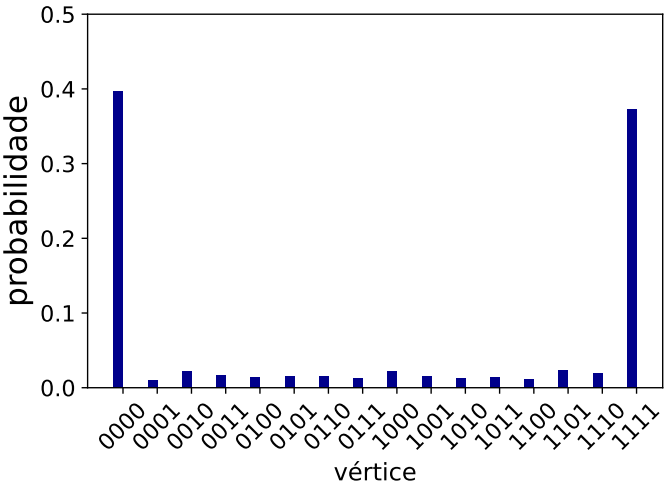

Fig. 5. Resultados da simulação com dois elementos marcados.

exibindo as distribuições de probabilidades onde, de fato, os elementos marcados $\omega_1 = 1111$ e $\omega_2 = 0000$ se destacam com as maiores ocorrências, conforme o esperado.

TABELA I
TEMPOS DE EXECUÇÃO EM SEGUNDOS COM 1 ELEMENTO MARCADO.

# qubits	CPU	GPU
4	0,7325	0,0382
5	0,6907	0,0419
6	0,6798	0,0390
7	0,6558	0,0461
8	0,6897	0,0441
9	0,6791	0,0419
10	0,6698	0,0468

TABELA II
TEMPOS DE EXECUÇÃO EM SEGUNDOS COM 2 ELEMENTOS MARCADOS.

# qubits	CPU	GPU
4	0,7148	0,0397
5	0,7257	0,0433
6	0,7204	0,0442
7	0,6827	0,0479
8	0,7020	0,0515
9	0,6799	0,0533
10	0,6738	0,0539

Os resultados apresentados nas Tabelas I e II demonstram claramente que a velocidade de execução em GPU é aproximadamente 12,7 vezes superior à da CPU. Isso ressalta a vantagem significativa do uso de GPUs, mesmo em circuitos de menor complexidade.

IV. CONCLUSÕES

- Podemos concluir que o uso de GPUs proporciona resultados significativamente mais rápidos em comparação com as CPUs. Essa velocidade é particularmente útil ao processar grandes volumes de dados e ao empregar um número maior de qubits em simulações.
- Considerando a atual escassez de computadores quânticos e as limitações das QPUs (Unidades de Processamento Quântico) no estágio NISQ (Noisy Intermediate-Scale Quantum), as placas gráficas (GPUs) se apresentam como uma alternativa viável e eficaz. Elas permitem simular sistemas complexos em cenários onde as QPUs existentes não seriam capazes de produzir resultados aceitáveis, especialmente ao implementarmos circuitos quânticos de maior escala.

AGRADECIMENTOS

Este trabalho foi financiado total pelo projeto QIN-AFCCT-FC-CC-2024-6-11-1 apoiado pelo QuIIN-Inovação Industrial Quântica, Centro de Competência EMBRAPII CIMATEC em Tecnologias Quânticas, com recursos financeiros do edital MCTI número 053/2023, firmado com a EMBRAPII.

REFERÊNCIAS

[1] C. Bennett. Logical reversibility of computation. *IBM Journal of Research and Development*, v. 17, n. 6, p. 525–532, 1973.
[2] T. Toffoli. Reversible computing. *In: 7th Colloquium on Automata, Languages and Programming*. Springer-Verlag. p. 632–644, 1980.
[3] P. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Review*, v. 41, 1999.
[4] L.Grover. Quantum Mechanics Helps in Searching for a Needle in a Haystack. *Phys. Rev. Lett*, v. 79, n. 325, 1997
[5] J. Preskill. Experimental realization of a momentum-space quantum walk. *Quantum 2*, v. 79, 2018.
[6] I. Chuang, N. Gershenfeld, M. Kubinec. Experimental Implementation of Fast Quantum Searching. *Physical Review Letters*, v. 80, n. 15, 1998.
[7] L. Grover. A fast quantum mechanical algorithm for database search. *ACM symposium on Theory of computing*, 1996.
[8] NVIDIA. CUDA Toolkit. Santa Clara, CA, [2024]. Disponível em: https://developer.nvidia.com/cuda-toolkit. Acesso em: 8 jul. 2025.
[9] NVIDIA. CUDA-Q. Santa Clara, CA, [2024]. Disponível em: https://developer.nvidia.com/cuda-q. Acesso em: 8 jul. 2025.