HybriD-GM: Um Modelo Paralelo para Computação Quântica direcionado às Arquiteturas Híbridas

Anderson Avila[1], Bruno Moura[2], Rafael Bastos[1], Helida Santos[3], Giancarlo Lucca[4], Anderson Cruz[5], Samuel de Xavier-de-Souza[5], Adenauer Yamin[1], Renata Reiser[1]

[1] PPGC – Universidade Federal de Pelotas (UFPel)
{abdavila,rrbastos,adenauer,reiser}@inf.ufpel.edu.br

[2]DTIC - Universidade Federal do Pampa (UNIPAMPA)
brunomoura@unipampa.edu.br

[3]C3 - Fundação Universidade de Rio Grande (FURG)
helida@furg.br

[4]PGEEC - Universidade Católica de Pelotas (UCPEL)
giancarlo.lucca@ucpel.edu.br

[5]DCA – Universidade Federal do Rio Grande do Norte (UFRN)
samuel@dca.ufrn.br

Abstract. *This paper has as its primary objective to introduce the the HybriD-GM model conception, as well as extend the D-GM environment, providing efficient parallel executions for quantum computing simulations, targeted to hybrid architectures considering both CPU and GPU. By managing projection operators over quantum structures, and exploring coalescing memory access patterns, the HybriD-GM model enables the granularity control, optimizing hardware resources in distributed computations organized as tree data-structures. In the HybriD-GM evaluation, simulations of Shor's and Grover's algorithms achieve significant performance improvements in comparison to the D-GM previous version and to the LIQUi$|\rangle$ and ProjectQ simulators.*

1. Introdução

A computação quântica (QC) promove soluções de alguns problemas que seriam intratáveis pela computação clássica. Assim, a perspectiva da supremacia quântica vem se tornando realidade, significando que um computador quântico pode realizar tarefas de cálculo que seriam inviáveis em um supercomputador clássico [Aaronson and Chen 2017].

Vários algoritmos quânticos (ordenação, fatoração de primos, exponenciação modular) já foram desenvolvidos, levando a melhorias significativas de desempenho quando comparados aos melhores algoritmos clássicos conhecidos. Em função disto, a computação quântica tem sido considerada potencialmente significativa em muitas áreas, como lógica reversível, criptografia, processamento de informações, comunicação, métodos de codificação de dados e muitas outras [Biswas et al. 2017].

Este trabalho tem por base o modelo HybriD-GM [de Avila et al. 2023], concebido como uma metodologia computacional para simulações de algoritmos quânticos em computadores clássicos. Por sua vez, o seu objetivo central é a concepção do modelo HybriD-GM, que explora as potencialidades da Computação de Alto Desempenho

(HPC) e se propõe reduzir o tempo necessário para as simulações de algoritmos quânticos, através de duas estratégias, as quais estão centradas em: (i) otimização de recursos e acesso à escalabilidade independente de hardware; e (ii) exploração de operadores de composição e projeção, incluindo o gerenciamento de memória coalescente.

Ambas as estratégias interpretam os Estados Quânticos (QS) e as Transformações Quânticas (QT) explorando o suporte oferecido pelas arquiteturas híbridas, integrando recursos computacionais do tipo CPU e GPU.

Os testes realizados quando da validação do HybriD-GM alcançaram melhorias significativas no desempenho quando da execução de algoritmos quânticos, tanto em relação à sua versão anterior, como no tocante a simuladores consolidados na literatura da área, como o LIQUi$|\rangle$ e o ProjectQ.

2. D-GM Framework e Trabalhos Relacionados

O framework D-GM, em desenvolvimento no grupo de pesquisa LUPS[1], é direcionado ao suporte do modelo qGM e a sua organização está apresentada na Fig. 1.

Nível de Circuito Quântico: descrevendo a aplicação no modelo de circuito com opção de exportação para representação usada no modelo qGM.

Nível qGM: contendo o Ambiente de Programação Visual para o Modelo qGM (VPE-qGM), que permite ao usuário descrever e disparar computações sob o *qGM*.

Nível D-GM: implementando o gerenciador de simulação distribuída denominado Máquina Geométrica Virtual Distribuída (VirD-GM), que lida com tarefas como agendamento, comunicação e sincronização necessárias em simulações distribuídas no qGM.

Nível de Hardware: gerenciando dispositivos acessados pelo framework, desde desktops regulares para simulações sequenciais, até mesmo *clusters* com múltiplas CPUs/GPUs.

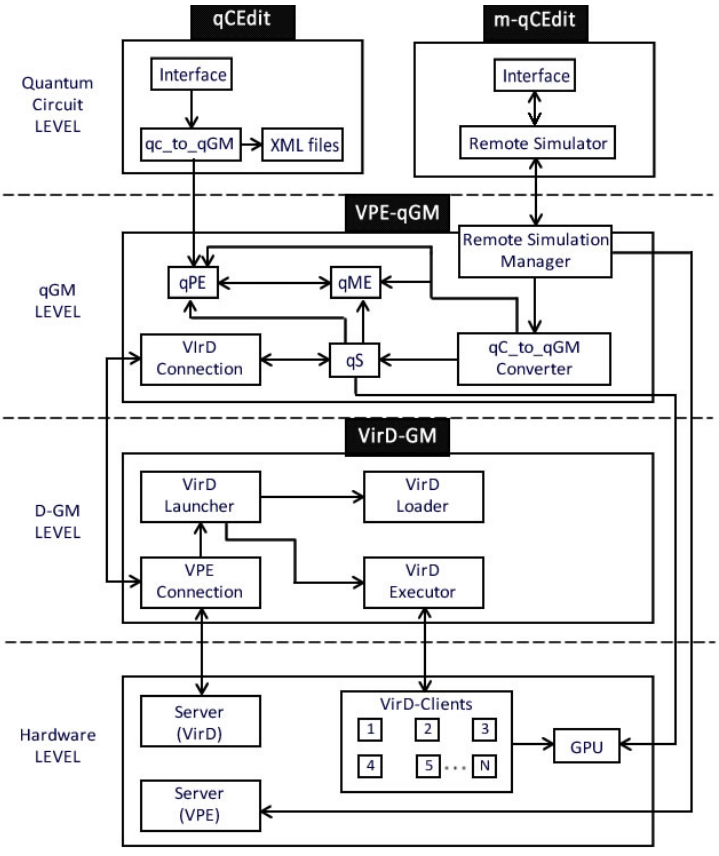


Figura 1. Níveis do Framework de Simulação D-GM.

Para seleção de Trabalhos Relacionados ao Framework D-GM foram considerados os seguintes aspectos:

(i) mesmo que em consolidação, trabalhos que lidem com a simulação de computação quântica de propósito geral;

[1]http://lups.inf.ufpel.edu.br

(ii) explorem recursos de HPC, com execuções envolvendo arquiteturas multi-core CPU, GPU e/ou distribuídas; e ainda,
(iii) disponibilizem relatórios das principais características, otimizações e resultados de simulação.

Considerando estes aspectos, a partir de uma revisão de literatura foram identificados seis trabalhos relacionados:

LIQUi|⟩[Wecker and Svore 2014], qHiPSTER[Smelyanskiy et al. 2016], ProjectQ [Steiger et al. 2016], Haner [Haner and Steiger 2017], Gutierrez [Gutierrez et al. 2010] e Zhang [Zhang et al. 2015]. Descrições e comparações, detalhadas estão disponíveis em [de Avila et al. 2023]. Resumindo, oportuno ressaltar que nenhum dos simuladores tem uma abordagem híbrida explorando o uso combinando de CPUs e GPUs.

Tendo por base este cenário, este trabalho propõe um modelo computacional para simulação de QC que pode ser aplicado a algoritmos de propósito geral, explorando simulações híbridas.

3. Proposta HybriD-GM: Modelo Conceitual

O espaço de estados e operadores em QC são descritos matematicamente pelo espaço de Hilbert, onde um registrador quântico composto por um número de qubits é representado como um vetor em um espaço de Hilbert multidimensional, enquanto portas quânticas são operadores de espaço de Hilbert que rotacionam os vetores do registrador quântico. Esta discussão está disponível na referência seminal [Hirvensalo 2001].

3.1. Computações com Projeções sobre Estados Quânticos

O modelo HybriD-GM proposto explora não apenas as características intrínsecas dos operadores de projeção geralmente aplicados a estados quânticos, mas também a dinâmica das computações baseadas em estruturas projetadas. Três das principais estratégias são detalhadas a seguir:

(I) Como uma projeção aplicada a uma estrutura de estado quântico implica em partições de seus componentes de base clássica, resulta em subconjuntos contendo suas amplitudes correspondentes. Veja, por exemplo, a Fig. 2 mostrando um estado genérico de 3 qubits, onde as amplitudes representadas por suas posições binárias são projetadas em sua segunda e terceira base em um processo de projeção em duas etapas.

(II) Sobre descrições de base para valores de memória aplicados em um operador de projeção:
WB : indicado por linhas de matriz sempre que o valor da base de escrita estiver associado à base do estado quântico que ele computa; e
RB : indicado por colunas de matriz, onde o valor RB está relacionado à base do estado quântico que ele usa para a computação.
Assim, operadores quânticos de um único qubit podem ser classificados em 3 tipos de acordo com seus valores não nulos, como mostrado na Fig. 3, gerando 4 pares de projeções de combinações entre as bases WB e RB. Assim, para operadores densos, cada WB está associado a dois RB e, para operadores esparsos, cada WB está associado a apenas um RB.

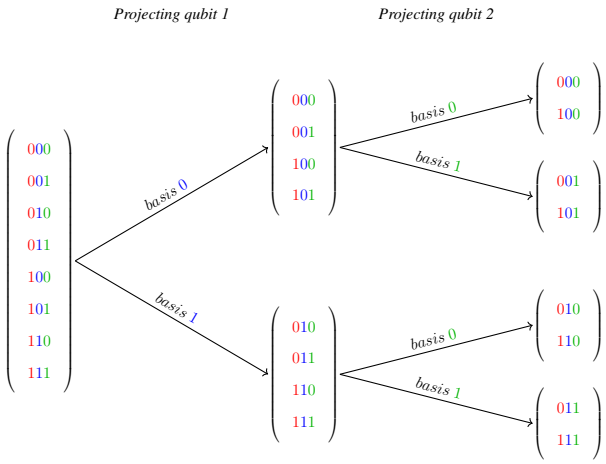


Figura 2. Projeções do 1º e 2º Qubits em um Estado Quântico Tridimensional.

Operators	Projections (WB,RB)			
$\begin{matrix} & 0 & 1 \\ 0 & e_{00} & e_{01} \\ 1 & e_{10} & e_{11} \end{matrix}$	0,0 (e_{00})	0,1 (e_{01})	1,0 (e_{10})	1,1 (e_{11})
$\begin{matrix} & 0 & 1 \\ 0 & e_{00} & 0 \\ 1 & 0 & e_{11} \end{matrix}$	0,0 (e_{00})	0,1 (0)	1,0 (0)	1,1 (e_{11})
$\begin{matrix} & 0 & 1 \\ 0 & 0 & e_{01} \\ 1 & e_{10} & 0 \end{matrix}$	0,0 (0)	0,1 (e_{01})	1,0 (e_{10})	1,1 (0)

Figura 3. Projeções de Operadores Quânticos

(III) Ao otimizar o produto tensorial, uma projeção de Transformação Quântica (QT) para uma determinada base pode ser obtida projetando-se individualmente o operador relacionado a essa base e, na sequência, realizando a multiplicação dos valores da projeção via produto tensorial entre entre os demais operadores. Por exemplo, seja uma QT bidimensional dado por $Id \otimes H$. Referente à projeção sobre a primeira base para o Id-operador, tem-se os resultados para $Id \otimes H$:

0,0	0,1	1,0	1,1
(1)	(0)	(0)	(1)

Assim, cada valor aplicado ao operador H resulta em uma das matrizes:

0,0	0,1	1,0	1,1
$\begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \end{pmatrix}$	$\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$	$\begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \end{pmatrix}$

3.2. Computações com Projeções sobre Estruturas Matriciais

Dado um Estado Quântico (QS) e um QT projetados em qubits/operadores (sobre bases relacionadas), a aplicação do QT neste QS computa a multiplicação entre cada projeção do QT e a projeção correspondente do QS (definindo um valor RB) e, em seguida, a soma dos resultados associados a um WB é executada, obtendo o QS resultante para tal base.

Assim, primeiramente as estruturas são projetadas e na sequência, têm-se as computações para reconstrução do QS, as quais são realizadas por operadores de imersão.

4. Proposta HybriD-GM: Modelo Arquitetural

O modelo HybriD-GM explora a projeção de QS e QT para controlar a distribuição e controle de granularidade das computações distribuídas, enquanto otimiza os recursos de hardware. Este trabalho foca em arquiteturas híbridas lidando com CPUs e GPUs, mas sua estrutura flexível permite extensões para outras arquiteturas. As estruturas de dados do HybriD-GM são organizadas em três níveis:

(1) **Estrutura de Projeção**, contendo amplitudes de QS que podem ser armazenadas direta ou indiretamente, e todas as informações necessárias para acesso e manipulação.
(2) **Instância de Projeção**, fornecendo uma referência à estrutura de projeção gerada, enumerando os valores da base projetada.
(3) **Estrutura de Porta**, contendo a construção matricial de QT, incluindo informações do qubit alvo e qubits/valores de controle, se houver.

Na camada de projeção genérica o módulo de controle solicita instâncias de projeção da estrutura de projeção, atuando nas seguintes ações:

(1) **Selecionar os qubits a serem projetados**, seguindo uma das seguintes abordagens: (i) a partir de intervalos que podem ser pré-definidos ou adaptados dos qubits alvo dos operadores, (ii) definindo dinamicamente conjuntos para incluir apenas qubits com operadores. Conjuntos fixos de qubits também podem ser incluídos na seleção em cada abordagem, o que permite projeções com amplitudes contíguas;
(2) **Projetar o estado quântico**, seja (i) direto - amplitudes correspondentes à instância de projeção dada são copiadas/transferidas para um novo espaço de memória, ou (ii) indireto - o estado de projeção, contendo as amplitudes, é passado como referência, com informações extras para determinar o acesso a essas amplitudes.
(3) **Projetar os operadores quânticos**, operadores atuando nos qubits selecionados são transferidos para uma sub-lista desde que possam ser executados preservando a consistência da computação.

Na Fig. 4, a estrutura arquitetônica do modelo HybriD-GM é apresentada graficamente e as características relevantes de seus cinco níveis são descritas na sequência:

Pré-processamento: recebendo os dados da aplicação para **decompor a aplicação quântica** em estruturas de portas e **converter o estado quântico** em uma estrutura de projeção.
Gerenciador de Projeção: gerenciando e executando n-projeções relacionadas às granularidades de computação. O módulo de controle é responsável por realizar as projeções em múltiplas camadas, usando os construtores SEQ e PAR para definir uma estrutura de dados em árvore, onde nós intermediários e finais são representados como camadas de projeção e camadas de execução, respectivamente.
Camadas de Projeção: fornecendo as configurações de projeção e identificando especificações sobre (i) granularidade, (ii) coalescência, (iii) forma de seleção dos qubits de projeção, e (iv) tipo de projeções de estado quântico para projeções tipo, considerando cada estrutura de hardware disponível nas computações.
Camadas de Execução: contendo os operadores de código e funções para computar projeções sobre cada estrutura de hardware disponível. E fornecendo camadas para otimizar a execução da CPU de portas quânticas de acordo com sua esparsidade, e camadas para gerenciar os dados para GPUs.
Hardware: estruturando o hardware alvo das execuções e integrando tanto multi-core e/ou multiprocessador quanto arquiteturas de CPU e/ou GPU.

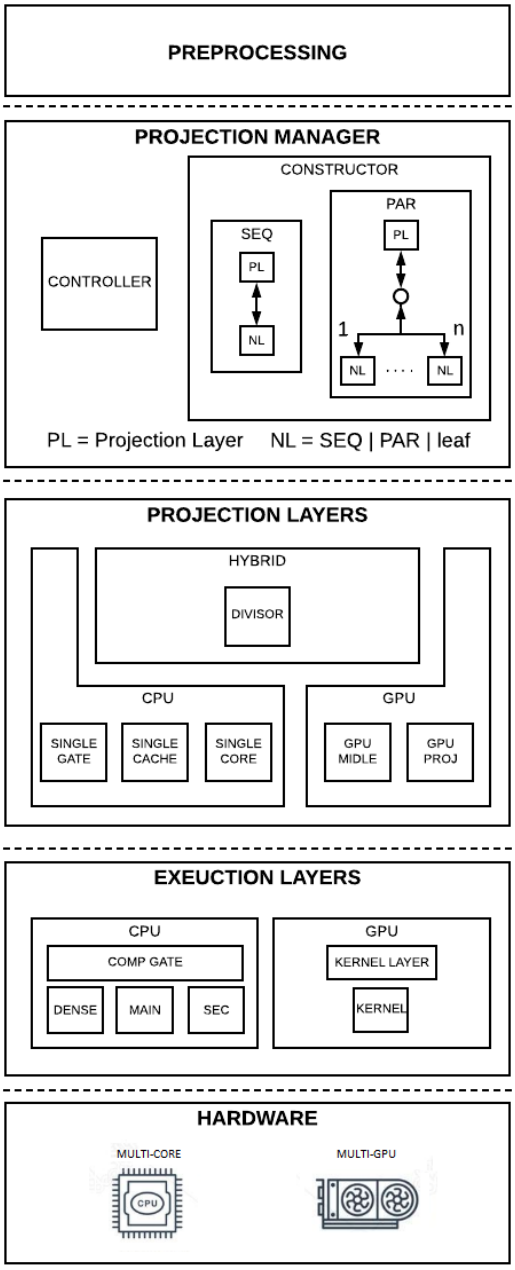


Figura 4. Arquitetura da Proposta HybriD-GM.

O gerenciador de projeções combina camadas de projeção e execução para simulações em CPU sequencial, CPU paralela, (MULTI-)GPU, bem como abordagens híbridas.

5. Proposta HybriD-GM: Estruturação de Projeções

O nível das camadas de projeção do modelo HybriD-GM contempla três categorias de projeções nos contextos de arquiteturas híbridas, relacionadas ao tipo de simulação: abordagens CPU, GPU, e Híbridas. A proposta visa granularidade, coalescência, forma de seleção dos qubits de projeção e tipo de projeções de estado quântico para otimizar os recursos de hardware para vários cenários, cujos requisitos são descritos a seguir.

1. Requisitos de projeção para CPU em simulações de CPU:
(i) **Single Gate**, considerando granularidade de 1 nível e sem coalescência, o qubit alvo

do primeiro operador na lista de estruturas de portas é usado para determinar o qubit de projeção e este operador também é projetado individualmente.
(ii) **Single Cache**, determinado pela granularidade e coalescência associadas ao estado quântico projetado, em relação ao espaço de memória e seus blocos de amplitudes subsequentes. Não pode exceder os tamanhos do último nível de cache para cada núcleo e o primeiro nível de cache, respectivamente. Utiliza abordagens fixas e dinâmicas para selecionar os qubits de projeção.
(iii) **Single Core**, onde a granularidade e a coalescência são sempre maiores do que as para Cache Único, com valores dependendo do tipo de simulação.

2. Requisitos de projeção para GPU em simulações de GPU:
(i) **GPU MIDDLE**, granularidade limitada para não exceder o tamanho da memória da GPU, e coalescência para ter o espaço de memória das amplitudes subsequentes igual ou maior do que o tamanho mínimo de transação de memória entre CPU e GPU.
(ii) **GPU PROJ**, granularidade definida para não exceder a memória compartilhada da GPU por bloco de threads, e coalescência para ter blocos de amplitudes subsequentes correspondentes ao tamanho da transação entre a memória global da GPU e a memória compartilhada.

3. Projeções híbridas permitem simulações híbridas em CPU e GPU, encapsulando os requisitos de projeção definidos acima, considerando a configuração baseado no conceito **DIVISOR**, onde a granularidade será um valor igual ou maior, e a coalescência, um valor igual aos valores máximos dos seus nós filhos.

O estado quântico na estrutura de projeção é indiretamente projetado para os cenários acima, exceto para a categoria GPU MÉDIA, que realiza uma projeção direta transferindo-o para a memória da GPU.

6. Proposta HybriD-GM: Abordagens de Execução

O Gerenciador de Projeção do modelo arquitetônico HybriD-GM, descrito na Seção 4, combina camadas de projeção e execução para possibilitar diferentes tipos de simulação [Avila et al. 2020].

Para todos os casos, as estruturas geradas pelos passos de pré-processamento são consideradas como entrada, significando que uma estrutura de projeção contendo uma única instância e uma lista de Estruturas de Portas representam, respectivamente, o estado quântico eleito e o algoritmo quântico para um processo de simulação. A simulação de núcleo único no modelo HybriD-GM pode ser observada na Figura 5(a), referida como EXECUÇÃO DE NÚCLEO ÚNICO, contendo quatro camadas descritas a seguir:

(i) SINGLE CACHE - camada que garante que as amplitudes acessadas para computar uma instância permanecerão no cache até o final da computação;
(ii) SINGLE GATE - garantindo cálculos realizados porta-por-porta;
(iii) COMP GATE - gerando instâncias ordenadas e, em seguida, garantindo a exploração da localidade espacial do nível de cache mais baixo, uma vez que cada par de amplitudes é subsequente ao par anterior;
(iv) EXE - referindo-se às camadas DENSE, MAIN e SEC de execução da CPU, dependendo do tipo de porta quântica.

O fluxo de simulação com abordagem GPU visto na Figura 5(b), e referido como EXECUÇÃO GPU, é realizado pelas seguintes camadas:

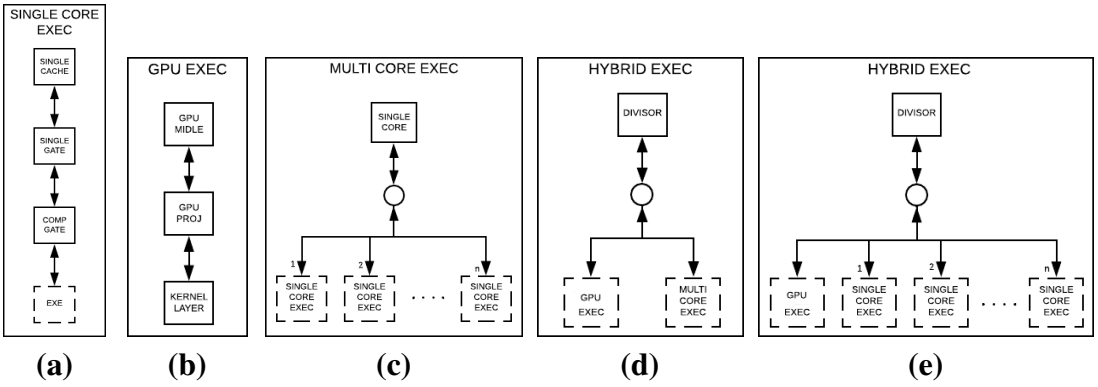


Figura 5. Composições de Níveis para as Simulações

- (i) GPU MIDDLE - camada que gera estruturas de projeção que correspondem ao tamanho da memória da GPU e, em seguida, as transfere para a memória da GPU;
- (ii) GPU PROJ - gera estruturas compatíveis com o tamanho da memória compartilhada por bloco de GPU;
- (iii) KERNEL LAYER - realiza chamadas de kernel para tarefas de computação.

A simulação na abordagem de execução multicore, mostrada na Fig. 5(c), é realizada em dois passos: (i) SINGLE CORE, onde as estruturas de projeção geradas são passadas para n nós, sendo n o número de núcleos utilizados na simulação; e ainda, (ii) SINGLE CORE EXEC, onde cada nó neste nível se refere a uma simulação de núcleo único, pois segue o mesmo fluxo de projeção/computações. Simulações quânticas com múltiplas GPUs seguem esta abordagem. No entanto, na primeira camada, o estado quântico da estrutura de projeção é transferido em partes iguais para cada GPU, e seus espaços de memória global devem ser visíveis por todos.

Existem duas abordagens para simulações híbridas, correspondendo a camadas distintas, consideradas em dois passos, conforme apresentado nas Figuras 5(d) e 5(e). Na camada DIVISOR, tem-se ambas as abordagens, fornecendo estruturas de projeção para seus nós filhos, que diferem em cada abordagem. A segunda camada apresenta: (i) um par de nós, GPU EXEC e MULTI CORE EXEC; e (ii) $n + 1$ nós filhos, GPU EXEC e n SINGLE CORE EXEC. E, as diferenças nas abordagens (i) e (ii) resultarão em granularidades distintas associadas a cada núcleo de CPU.

7. Proposta HybriD-GM: Avaliação de Simulações das Aplicações

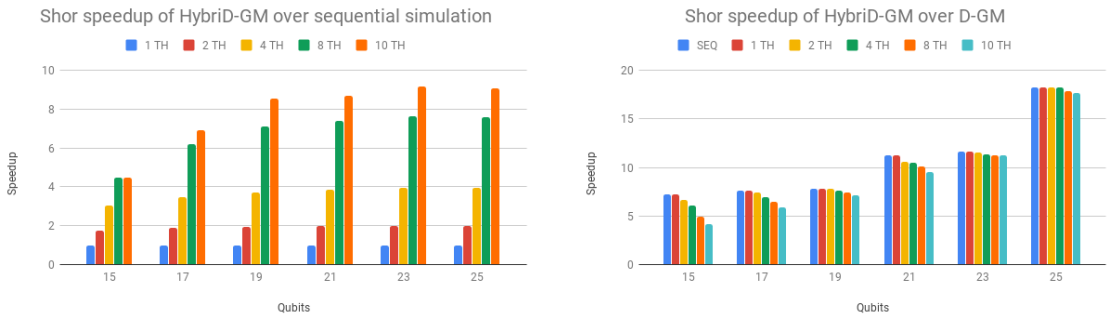
Os resultados consideram testes foram realizados em desktop com um processador Intel Core i9-7900X com dez núcleos, 32 GB de RAM e duas GPUs NVidia GTX Titan X. Os experimentos foram executados no Ubuntu Linux versão 17.04, 64 bits, e CUDA Toolkit 9.0. Os tempos médios de simulação foram calculados a partir de 30 execuções, e todos os casos apresentaram um desvio padrão menor que 1%.

7.1. Simulações via CPU

Para otimizar as simulações em CPU, as camadas de projeção consideram o tamanho da cache disponível na arquitetura alvo, no caso das simulações deste trabalho (L2 - 1024Kb, L1 - 32Kb), o uso de 16 qubits (512Kb) foi definido para o tamanho da estrutura de

projeção considerando uma coalescência de 11 qubits (16Kb) para otimizar o uso dos níveis de cache (evitando o uso completo da cache).

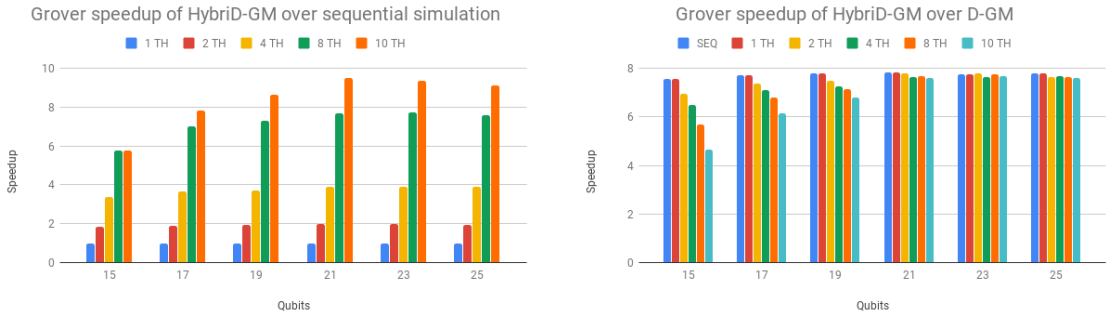
Os algoritmos de Shor e Grover foram simulados sequencialmente e em paralelo na CPU usando o framework D-GM [Avila et al. 2019], e a extensão HybriD-GM, variando o número de threads de 1 a 10 e considerando uma faixa de qubits de 15 a 25.



(a) Em Relação à Simulação Sequencial. **(b) Em Relação à Versão Anterior.**

Figura 6. Speedup do Algoritmo de Shor na Simulação em CPU do HybriD-GM.

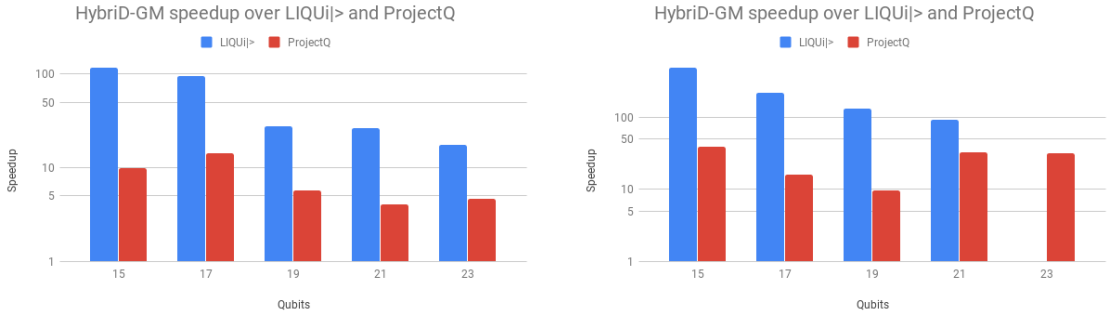
Veja na Fig. 6 os speedups das simulações do algoritmo de **Shor** no HybriD-GM. Em (a), são apresentados os speedups em relação à versão sequencial e ganho de desempenho para todas as simulações com múltiplas threads, mostrando melhor escalabilidade conforme o aumento do número de qubits. E assim, o overhead das computações não paralelas se torna menos significativo, atingindo um ganho de $9,18\times$ para uma simulação de 21 qubits. E, em (b) são mostrados os speedups em relação versão anterior, com desempenho superior para todas as simulações, aumentando com o número de qubits até alcançar um desempenho $18\times$ mais rápido para simulações de 25 qubits. Salienta-se



(a) Em relação à Simulação Sequencial. **(b) Em Relação à Versão Anterior.**

Figura 7. Speedup do Algoritmo de Grover na Simulação em CPU do HybriD-GM.

ainda os speedsups obtidos para as simulações do algoritmo de **Grover** no HybriD-GM: (a) são apresentados os speedups em relação à versão sequencial, com ganho de desempenho para todas as simulações com múltiplas threads, e com escalabilidade semelhante ao algoritmo de Shor, alcançando até $9.51\times$ para a simulação de 21 qubits; (b) é apresentado o speedup do HybriD-GM sobre o D-GM, com uma escalabilidade constante para simulações de 21 qubits com um ganho de desempenho de cerca de $7.9\times$.



(a) Shor **(b) Grover**

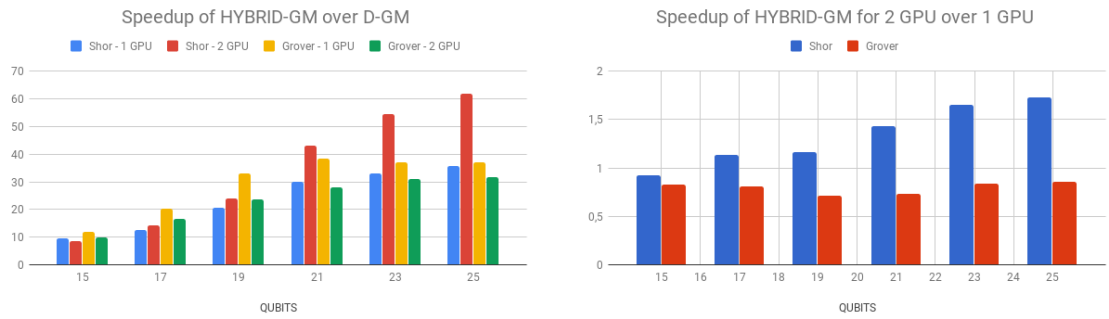
Figura 8. Speedup do HybriD-GM com 10 Threads sobre LIQUi$|\rangle$e ProjectQ.

Os ganhos de velocidade sobre LIQUi$|\rangle$ e ProjectQ são mostrados na Fig. 8, usando uma escala logarítmica para melhor visualização. Pode-se observar que o D-GM apresentou um desempenho melhor em todos os cenários, com ganhos maiores para simulações de poucos qubits, isto é decorrência do D-GM possuir um overhead menor, e conforme o aumento do número de qubits este ganho se estabiliza. Desconsiderando os resultados aonde o principal fator foi o overhead, o algoritmo de Shor mostrou ganho de $17,7\times$ em relação ao LIQUi$|\rangle$ e $14,2\times$ em relação ao ProjectQ, e o algoritmo de Grover mostrou ganho de $14,2\times$ em relação ao LIQUi$|\rangle$ e $9.7\times$ em relação ao ProjectQ.

7.2. Simulações via GPU

Para otimizar as simulações foi considerado o fator de coalescência de 4 qubits, pois é o valor mínimo para garantir que todas as amplitudes acessadas em cada requisição de memória pertençam à mesma projeção.

Na Fig. 9 tem-se a relação de desempenho do HybriD-GM sobre o D-GM em duas situações: (a) mostrando melhorias para 2 GPUs e 1 GPU, até $38,32\times$ e $61,90\times$. Por sua vez, na Fig. 9 (b) são comparadas simulações multi-GPU no modelo HybriD-GM. Observou-se no caso Multi-GPU que somente o algoritmo de Shor conseguiu ganho de desempenho sendo até $1.73\times$, enquanto o algoritmo de Grover não mostra ganho significativo, pois grande parte de operadores possuem controles nos qubits que foram projetados. Logo, uma única GPU recebeu a maioria das computações efetivas.



(a) Em Relação ao D-GM. **(b) Multi-GPU.**

Figura 9. Speedup das Simulações em GPU do HybriD-GM.

7.3. Simulações Hibridas

As execuções dos algoritmos de Shor e Grover, cujos resultados estão reportados na Tabela 1, foram realizadas para uma simulação de 25 qubits, limitando a memória da GPU a 20 qubits.

Observa-se que simulações apenas com GPU aumentaram o tempo para ambos os algoritmos, em contraste dos tempos sem limitação de memória. Uma vez que a GPU é muito mais eficiente para realizar a computação das projeções, o principal fator que impacta o resultado da simulação hibrida é o tempo a GPU tende a ficar ociosa esperando pelo término das computações em CPU para poder realizar a sincronização e prosseguir para a próxima projeção. Este tempo ocioso tende a ser maior quanto mais operadores uma projeção possui, impactando no tempo de computação de uma projeção.

No algoritmo de Shor pode-se observar que o desempenho diminuiu com o uso da CPU enquanto o algoritmo de Grover foi até $1,48\times$ mais rápido, isto ocorre por que o número médio de operadores/projeção do algoritmo de Shor é relativamente maior e logo as computações realizadas pela CPU não compensaram o tempo ocioso da GPU.

Tabela 1. Simulação para uma Execução Híbrida dos Algoritmos de Shor e Grover com 25 Qubits, Limitada a 20 Qubits para a Memória da GPU.

	Apenas GPU	1 Thread	2 Threads	4 Threads	8 Threads	10 Threads
Shor	74.40	137.09	136.63	140.91	157.81	171.69
Grover	1023.20	1247.79	1202.79	1106.32	690.52	848.71

8. Conclusão

A simulação de CQ mostra-se como uma aplicação interessante para HPC, à medida que a complexidade espacial e temporal aumenta exponencialmente nas dimensões dos qubit das simulações. Este trabalho contribuiu para o desenvolvimento do modelo HybriD-GM, explorando operadores de projeção atuando em estruturas quânticas como QS e QT, manipulando não apenas dados de memória coalescidos, mas principalmente a granularidade e distribuição de computações quânticas.

A estratégia HybriD-GM para desempenho computacional é modelada como uma estrutura de dados em árvore, onde os nós intermediários e finais representam camadas de projeção e execução, respectivamente. Tal estrutura está configurada para otimizar os recursos de hardware para os cenários propostos, que no estado atual do modelo são baseados em simulações em CPU, GPU e abordagens híbridas. As otimizações discutiram operadores de projeção e a relação em termos de decomposição de produtos tensoriais via QT e QS.

O modelo HybriD-GM suporta novas extensões, permitindo a adição de etapas de otimizações, cenários de projeção/computação e outros tipos de simulações. Além disso, a arquitetura de software híbrida para computação quântica é concebida como independente de hardware, onde os cálculos podem ser realizados por desktops regulares para simulações sequenciais de aplicações quânticas multi-qubits ou por clusters com múltiplas GPUs.

Trabalhos futuros no modelo HybriD-GM consistem de vários tópicos, incluindo: (i) melhorias na distribuição das projeções em simulações híbridas à fim de diminuir a

ociosidade da GPU/CPU, (ii) consolidação do modelo realizando simulações com um maior número de qubits e em super computadores, (iii) e também buscar o incremento da simulação de algoritmos fuzzy baseados em computação quântica.

9. Agradecimentos

Os autores agradecem as seguintes agências de fomento pelos projetos: CNPq (309559/2022-7, 409696/2022-6), PqG/FAPERGS (21/2551-0002057-1), PqG/FAPERGS (24/2551-0001396-2), ARD-ARC/FAPERGS (24/2551-0000631-1) e FAPERGS/CNPq (23/2551-0000126-8).

Referências

Aaronson, S. and Chen, L. (2017). Complexity-Theoretic Foundations of Quantum Supremacy Experiments. In O'Donnell, R., editor, *32nd Computational Complexity Conference (CCC 2017)*, volume 79 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 22:1–22:67, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

Avila, A., Reiser, R., Pilla, M., and Yamin, A. (2019). Improving in situ GPU simulation of quantum computing in the D-GM environment. *Int. J. High Perform. Comput. Appl.*, 33(3).

Avila, A., Reiser, R., Pilla, M., and Yamin, A. (2020). State-of-the-art quantum computing simulators: Features, optimizations, and improvements for d-gm. *Neurocomputing*, 393:223 – 233.

Biswas, R., Jiang, Z., Kechezhi, K., Knysh, S., Mandr, S., OGorman, B., Perdomo-Ortiz, A., Petukhov, A., Realpe-Gmez, J., Rieffel, E., Venturelli, D., Vasko, F., and Wang, Z. (2017). A nasa perspective on quantum computing. *Parallel Comput.*, 64(C):81–98.

de Avila, A. B., Santos, H. S., Cruz, A. P., de Souza, S. X., Lucca, G., Moura, B., Yamin, A. C., and Reiser, R. (2023). Hybrid-gm: A framework for quantum computing simulation targeted to hybrid parallel architectures. *Entropy*, 25(3):503.

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

Haner, T. and Steiger, D. S. (2017). 0.5 petabyte simulation of a 45-qubit quantum circuit. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '17, New York, NY, USA. Association for Computing Machinery.

Hirvensalo, M. (2001). *Quantum Computing*. Springer Verlag, Natural Computing Series.

Smelyanskiy, M., Sawaya, N. P. D., and Aspuru-Guzik, A. (2016). qHiPSTER: The Quantum High Performance Software Testing Environment.

Steiger, D. S., Häner, T., and Troyer, M. (2016). ProjectQ: An open source software framework for quantum computing.

Wecker, D. and Svore, K. M. (2014). Liqui$|>$: A software design architecture and domain-specific language for quantum computing. *Computing Research Repository (CoRR)*, abs/1402.4467.

Zhang, P., Yuan, J., and Lu, X. (2015). *Quantum Computer Simulation on Multi-GPU Incorporating Data Locality*, pages 241–256. Springer International Publishing, Cham.