Otimizando a Implementação Multi-GPU do Método Fletcher através da Paralelização Eficiente na Computação e Comunicação de Dados *

Pedro H. C. Rigon[1], **Brenda S. Schussler**[1], **Cristiano A. Künas**[1], **Arthur F. Lorenzon**[1], **Alexandre Carissimi**[1], **Philippe O. A. Navaux**[1]

[1]Instituto de Informática – UFRGS - Porto Alegre – RS – Brazil

{phcrigon, bsschussler, cakunas, aflorenzon, asc, navaux}@inf.ufrgs.br

Resumo. *Este estudo explora a otimização de um sistema Multi-GPU para a aplicação do Método de Fletcher na exploração geofísica visando aprimorar as aplicações que envolvem esse tipo de modelagem. Com a crescente disponibilidade de várias GPUs em servidores de alto desempenho, nosso foco é otimizar a implementação multi-GPU do Método Fletcher, possibilitando a computação e comunicação de dados de forma paralela. Essa abordagem não apenas assegura uma resolução eficiente da Equação Diferencial Parcial que modela a propagação de ondas sísmicas, mas também aprimora a eficiência global no processamento através da redução do overhead de comunicação, resultando em uma redução de até 40% no EDP (Energy-Delay Product).*

1. Introdução

A busca por recursos como petróleo e gás, por meio da exploração geofísica, tem desempenhado um papel fundamental no desenvolvimento econômico global. Entretanto, as práticas tradicionais associadas à identificação de novos reservatórios frequentemente adotam métodos intrusivos, como perfurações em regiões ambientalmente sensíveis, resultando em impactos adversos substanciais.Para lidar com esses desafios e aprimorar a precisão da exploração, a comunidade cientifica tem explorado aplicações que simulam imagens sísmicas para detecção de petróleo, destacando a importância das Unidades de Processamento Gráfico (GPUs) devido a sua capacidade de processamento [Navaux et al. 2023].

Com a crescente disponibilidade de múltiplas GPUs em servidores de alto desempenho, há a oportunidade de explorar de maneira mais abrangente o potencial de processamento destes sistemas. Nesse contexto, torna-se necessário não apenas otimizar a computação paralela em cada GPU, mas também, melhorar a eficiência na troca de dados entre as GPUs. Uma vez que ao lidar com ambientes multi-GPU, a minimização do movimento de dados entre dispositivos e a implementação de estratégias inteligentes de paralelização são fundamentais para reduzir custos associados à computação e mitigar os gargalos causados pelo movimento de dados, visando um desempenho global mais eficiente [Liu et al. 2019].

Neste artigo, propomos uma otimização na implementação Multi-GPU do método Fletcher, onde a computação e a comunicação de dados entre as GPUs ocorrem de forma

*Este estudo foi parcialmente apoiado pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001, pela Petrobras sob número 2020/00182-5 e pelo edital CNPq/MCTI/FNDCT - Universal 18/2021 sob número 406182/2021-3. Alguns experimentos deste trabalho utilizaram os recursos da infraestrutura PCAD, http://gppd-hpc.inf.ufrgs.br, no INF/UFRGS.

paralela. Esta abordagem não apenas visa uma resolução eficiente da Equação Diferencial Parcial, que modela a propagação das ondas sísmicas no tempo, mas também visa atingir ganhos de EDP (*Energy-Delay Product*) através da redução do *overhead* de comunicação inter-GPU, resultando em uma redução de até 40% no EDP.

2. Método Fletcher

O método Fletcher modela a propagação de ondas acústicas em um ambiente *Tilted Transversely Isotropic (TTI)* por meio de uma *Grid* tridimensional. Cada ponto nessa *Grid* representa um ponto no meio físico sendo modelado, e esse ponto está associado a propriedades físicas como pressão, densidade e velocidade da onda. Na implementação Multi-GPU deste método, proposta em [Rigon et al. 2023], a distribuição da carga de trabalho é realizada através da divisão do eixo z entre todas as GPUs, como ilustrado na Figura 1, para a distribuição entre 4 GPUs.

Além disso, cada ponto da *Grid* é computado de forma independente dos demais. No entanto, a propagação da onda ocorre através do cálculo de um *Stencil* de 5 pontos, baseado na leitura dos pontos vizinhos em cada eixo. Dessa forma, a variável que influencia esse cálculo refere-se ao estado anterior da onda, e essa variável não é atualizada durante a execução do *kernel* [Pearson et al. 2020], não havendo dependência de dados entre as threads em execução. Entretanto, durante a execução do *kernel*, é necessário atualizar as bordas do domínio responsáveis pela informação atual da onda, tal qual representado na Figura 2, que são as regiões de leitura da próxima iteração do *kernel*, e, portanto, requerem atualização para gerar resultados consistentes.

3. Otimização Proposta

Considerando o discutido anteriormente, paralelizar a atualização das regiões de borda com a computação do restante da *Grid* é uma alternativa de otimização. Para isso, inicialmente, cada GPU realiza a computação da borda que será atualizada em sua respectiva GPU vizinha, que seria a GPU que contém uma cópia desatualizada da região de borda e requer comunicação inter-GPU para atualização. Após o término da computação da borda, será criado um *stream CUDA* exclusivamente para a comunicação inter-GPU tal qual demonstrado na Figura 3, paralelamente ao *stream* principal, onde ocorrerá a computação do restante da *Grid*. Vale ressaltar que essa comunicação inter-GPU é do tipo P2P *(Peer-to-Peer)*, o que significa que as GPUs se comunicam diretamente entre si, sem a necessidade de passar pela *CPU (Central Processing Unit)* ou pela memória do sistema, proporcionando uma troca de dados mais eficiente e uma redução na latência.

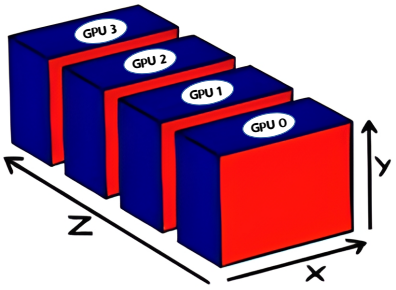


Figura 1. Distribuição de carga de trabalho ao longo do eixo z entre 4 GPUs

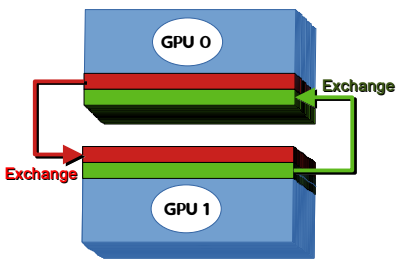


Figura 2. Atualização das regiões de borda através da comunicação inter-GPU.

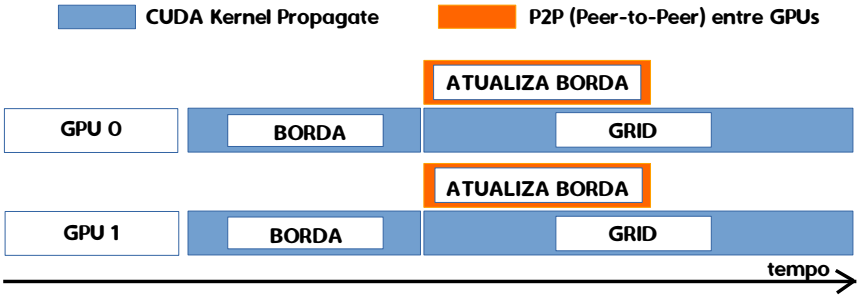


Figura 3. Simulação de um profiling da aplicação, destacando o paralelismo entre a comunicação inter-GPU e a computação da grid.

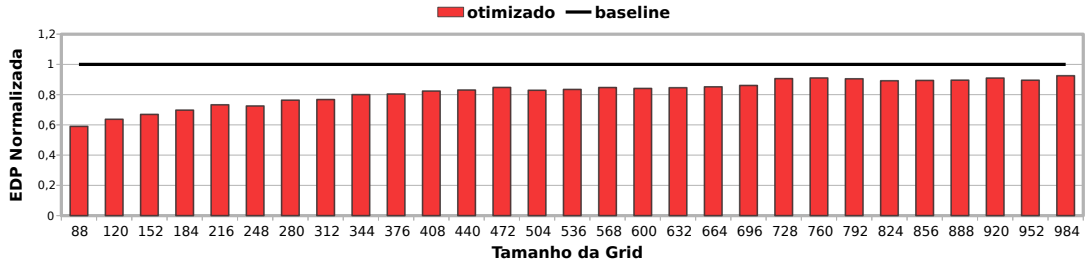
4. Resultados

Os experimentos foram realizados utilizando a implementação multi-GPU otimizada do Método Fletcher em CUDA para 2 GPUs e 4GPUs. Os resultados apresentados foram obtidos através da média de 10 execuções com intervalos de confiança de 95% segundo a distribuição *T-Student*. As execuções foram realizadas no Parque Computacional de Alto Desempenho da UFRGS (PCAD) utilizando uma arquitetura Pascal com 4 GPUs NVIDIA Tesla P100-SXM2-16GB, permitindo até 3584 núcleos CUDA, com dois processadores Intel Xeon E5-2699, cada um com 22 núcleos. As versões utilizadas foram CUDA v.11.8, driver NVIDIA 525.85.12 e GCC 9.4.0 com a flag -O3.

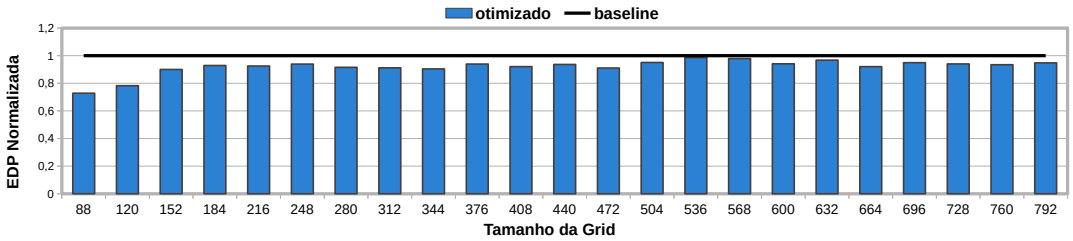
Os resultados são representados através dos gráficos de EDP (*Energy-Delay Product*), pois define o custo-benefício entre desempenho e consumo de energia dando uma visão mais abrangente do resultado. Os valores são normalizados em relação ao *baseline*, que seria a implementação proposta em [Rigon et al. 2023], representado pela linha horizontal em 1. Valores abaixo representam ganho em EDP. Portanto, quanto menor o valor em relação ao *baseline*, maior é o ganho de EDP. Vale ressaltar que as execuções com 4 GPUs permitem um conjunto de *Grids* maior (*88 - 984*) em comparação a execução com 2 GPUS (*88 - 792*), devido à maior disponibilidade Global de VRAM *(Video Random Access Memory)*.

Após análise dos gráficos, observamos ganhos consideráveis tanto para 4 GPUs (Figura 4(a)), quanto para 2 GPUs (Figura 4(b)). A otimização teve um impacto mais evidente nas execuções com *Grids* menores. Devido ao fato de que, para esses tamanhos de *Grids*, o custo associado à comunicação da borda do domínio computacional representa uma parte significativa do tempo de execução. Portanto, com a nossa otimização proposta, esse custo foi mitigado pela paralelização da comunicação inter-GPU com a computação da parte útil da *Grid*, gerando ganhos em relação à versão *baseline*. Além disso, à medida que o tamanho da *Grid* aumentou, observamos uma estabilização nessa diferença. Isso, possivelmente, se deve ao aumento do custo computacional em relação ao custo de comunicação, porém, ainda mantendo ganhos consideráveis em comparação com o *baseline*.

Outro aspecto a ser observado é que, ao aumentar o número de GPUs, maior é o número de bordas do domínio computacional. Por exemplo, entre 2 GPUs e 4 GPUs, o número de bordas do domínio por GPU dobra, e, portanto, é razoável inferir que que o custo de comunicação dessas regiões será maior. Essa hipótese está em conformidade com os resultados apresentados, uma vez que o gráfico com 4 GPUs mostrou uma melhora comparativamente maior do que a execução com 2 GPUs, indicando que o impacto da



(a) Resultados utilizando **4 GPUs**.



(b) Resultados utilizando **2 GPUs**.

Figura 4. Resultados de EDP normalizados para a otimização do método Fletcher multi-GPU em comparação com *baseline* (sem otimização) para 4 e 2 GPUs.

otimização é mais evidente ao escalar o número de GPUs.

5. Conclusão

Com base nos resultados apresentados, podemos concluir que a otimização da implementação multi-GPU do método Fletcher trouxe ganhos significativos, resultando em uma redução de até 40% no EDP *(Energy-Delay Product)*. A paralelização da comunicação inter-GPU e do cálculo da *Grid* demonstrou ser uma estratégia eficaz, especialmente em execuções com *Grids* menores, onde o custo de comunicação na borda do domínio computacional é mais significativo. Além disso, ao aumentar o número de GPUs, observamos uma melhoria proporcional no EDP. Como trabalhos futuros, planejamos avaliar o impacto de diferentes distribuições de carga de trabalho para o método Fletcher, além de estimar de maneira mais precisa o impacto da comunicação de dados inter-GPUs no desempenho da aplicação.

Referências

Liu, G.-F., Meng, X.-H., Yu, Z.-J., and Liu, D.-J. (2019). An efficient scheme for multi-GPU TTI reverse time migration. *Applied Geophysics*, 16(1):56–63.

Navaux, P. O. A., Lorenzon, A. F., and da Silva Serpa, M. (2023). Challenges in High-Performance Computing. *Journal of the Brazilian Computer Society*, 29(1):51–62.

Pearson, C., Hidayetoğlu, M., Almasri, M., Anjum, O., Chung, I.-H., Xiong, J., and Hwu, W.-M. W. (2020). Node-aware stencil communication for heterogeneous supercomputers. In *2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 796–805. IEEE.

Rigon, P. H., Schussler, B. S., Padoin, E. L., Lorenzon, A. F., Carissimi, A., and Navaux, P. O. (2023). Towards a Multi-GPU Implementation of a Seismic Application. In *Latin American High Performance Computing Conference*, pages 146–159. Springer.