Aceleração de métodos de empilhamento de dados sísmicos na nuvem com CUDA, OpenCL e SPITS

Gustavo Ciotto Pinton[1], **Edson Borin**[1]

[1]Instituto de Computação – Universidade Estadual de Campinas (UNICAMP)
Av. Albert Einstein, 1251 - Cidade Universitária, Campinas - SP, 13083-852

Abstract. *Seismic processing algorithms have been important in many industry applications, notably oil and gas exploration. Such methods tend to be computationally expensive mainly due to the big amount of data. In this article, we evaluate an application to search for the environment parameters that maximizes the coherence measure for three different traveltime estimation models (including a brand new implementation for the OCT model), which automatically distributes idempotent and independent tasks to CUDA/OpenCL supporting nodes of a computing cloud. Besides, to avoid performance degradation due to phenomena such as data transfer and cache misses, we introduce a heuristic to select which fraction of the data should be indeed considered.*

Resumo. *Técnicas de imageamento do subsolo marítimo vêm sendo fundamentais para diversas aplicações da indústria, notadamente para a exploração de petróleo e gás. Neste artigo, nós avaliamos uma implementação para a procura de parâmetros do meio marítimo maximizando a medida de coerência para três diferentes modelos de tempo de trânsito 2D (incluindo o modelo OCT) que automaticamente realiza a distribuição de tarefas idempotentes e independentes a nós de uma nuvem computacional com suporte às plataformas CUDA e OpenCL. Além disso, para evitar degredação de desempenho devido a fenômenos como a transferência de dados e* cache misses*, nós introduzimos uma heurística para a seleção da fração dos dados que deve ser de fato considerada.*

1. Introdução

No dia 6 de novembro de 2019, o Brasil alcançou uma marca histórica ao arrecadar perto de 70 bilhões de reais no leilão das bacias de petróleo Búzios e Itaipu pertencentes à cessão onerosa do pré-sal. Tal valor é tão expressivo que chega a superar a soma acumulada por todos os outros demais leilões na história do país [Mendonça et al. 2019], além de ser importantíssimo para o seu desenvolvimento social e tecnológico.

Antes de quaisquer intervenções humanas em solo marítimo, a exploração de novas fontes de energia requer um entendimento prévio e detalhado da sua topologia e dos principais parâmetros que caracterizam o meio, tendo em vista que muitos outros investimentos dependerão dos resultados obtidos por este mapeamento. Neste cenário, alguns modelos foram elaborados para equacionar o tempo de trânsito de um sinal sonoro emitido por uma fonte e captado por seu respectivo receptor, após ter viajado neste meio e sido refletido pelo solo marinho. Tais modelos dependem essencialmente dos parâmetros do meio, da disposição física (m, h) de um par fonte-receptor e de um ponto (t_0, m_0, h_0) de referência, para o qual se quer computar efetivamente a aproximação [Faccipieri et al. 2019]. Na sentença anterior, m e h representam, respectivamente, o ponto médio entre uma fonte e o seu receptor, e a metade da distância que o separam. Sendo assim, os métodos de empilhamento visam determinar os parâmetros

$P(t_0, m_0, h_0)$ que melhor se adequam aos dados adquiridos para cada instante de tempo t_0 e posição (m_0, h_0) a partir de uma métrica denominada de *semblance*.

Para este trabalho, escolhemos três equações de tempo de trânsito distintas com o intuito de avaliarmos se a solução proposta é capaz de lidar com grandes volumes de dados para diversos níveis de complexidade. Tais equações são, em ordem crescente de número médio de dados usados no cálculo de cada $P(t_0, m_0, h_0)$, relacionadas aos métodos *Common Mid Point*, *Zero Offset Common Reflection Surface* [Faccipieri 2012] e *Offset Continuation Trajectory* [Coimbra et al. 2012].

É importante destacar que a procura pelo melhor parâmetro para cada instante de tempo e posição é totalmente paralelizável, isto é, o cálculo de *semblance* para o parâmetro $P_0(t_0, m_0, h_0)$ é totalmente independente daquele baseado em $P_1(t_1, m_1, h_1)$, mesmo se $t_0 = t_1$ e $(m_0, h_0) = (m_1, h_1)$ [da Silva 2017][Gimenes et al. 2018].

Neste trabalho, propomos a implementação da procura dos parâmetros $P(t_0, m_0, h_0)$ que melhor se adequem aos dados de entrada, a partir do método de empilhamento e de acordo com a métrica de *semblance*, com suporte à paralelização de tarefas em aceleradores compatíveis com os *frameworks* CUDA e/ou OpenCL em diversos nós de uma rede computacional através do *SPITS* [Borin et al. 2015]. Além disso, a fim de otimizar a quantidade de parâmetros a serem utilizados durante a busca para cada (t_0, m_0, h_0), escolhemos empregar a técnica de computação evolutiva denominada de *evolução diferencial* [Barros et al. 2015]. Adicionalmente, propomos uma heurística para otimizar a quantidade de dados a serem transmitidos ao acelerador.

2. Seleção dos traços

Uma parte fundamental da implementação é a seleção dos traços que farão parte do conjunto de empilhamento $\mathbb{S}$ para cada ponto (t_0, m_0, h_0). Além de diminuir a quantidade de dados a serem transmitidos à GPU, ao selecionar apenas os traços que serão utilizados de fato, reduzimos a complexidade do *kernel* que será executado. A seleção de traços também busca evitar que desvios de fluxo ocorram apenas em parte das *threads* de um bloco.

A seleção de um traço para os modelos *Common Mid Point* e *Zero Offset Common Reflection Surface* é trivial já que depende apenas de m_0 e do ponto médio m_f deste próprio traço. Por outro lado, como a seleção de traços para o modelo *Offset Continuation Trajectory* depende de $P(t_0, m_0, h_0)$, adotamos a seguinte heurística: consideramos o conjunto $\mathbb{P} = \{P(t_0, m_0, h_0) \mid \forall (t_0, m_0, h_0)\}$ contendo todos os parâmetros para os quais a *semblance* será calculada e inserimos o traço i ao conjunto $\mathbb{S}$ se ele for utilizado por, pelo menos, um $P(t_0, m_0, h_0) \in \mathbb{P}$. Para o caso da evolução diferencial, em que as populações são iniciadas de maneira uniforme sobre todo o intervalo de busca, os resultados de nossos testes preliminares demonstraram que $\mathbb{P}$ não cresce ao decorrer das gerações. Desta forma, podemos executar a seleção apenas uma vez antes que a primeira geração seja calculada.

3. Resultados

3.1. Dados sísmicos de teste

Buscamos utilizar dados de entrada apresentando bastante variância tanto no número de amostras por traço quanto na quantidade de traços. Utilizamos, até o momento, três dados gerados sinteticamente, sendo **simple-synthetic**, **fold1000** e **fold2000**. Tais dados apresentam, respectivamente, 6×10^3, 4×10^5 e 8×10^5 traços com 2502, 751 e 751 amostras cada.

	$\mathbb{S}_{CMP}$		$\mathbb{S}_{all}$	
	$\overline{\eta_{eff}}$	$\overline{r_{intr}}$	$\overline{\eta_{eff}}$	$\overline{r_{intr}}$
simple-synthetic	1	1.56×10^{10}	0.0023	7.3×10^{8}
fold1000	1	1.86×10^{10}	0.0025	3.42×10^{8}
fold2000	1	1.86×10^{10}	0.0025	3.42×10^{8}

Tabela 1. Resultados para T_{CMP}.

	$\mathbb{S}_{ZOCRS}$		$\mathbb{S}_{all}$	
	$\overline{\eta_{eff}}$	$\overline{r_{intr}}$	$\overline{\eta_{eff}}$	$\overline{r_{intr}}$
simple-synthetic	1	1.38×10^{10}	0.034	6.24×10^{9}
fold1000	1	1.55×10^{10}	0.032	3.43×10^{9}
fold2000	1	1.54×10^{10}	0.032	3.43×10^{9}

Tabela 2. Resultados para T_{ZOCRS}.

	$\mathbb{S}_{OCT}$		$\mathbb{S}_{all}$	
	$\overline{\eta_{eff}}$	$\overline{r_{intr}}$	$\overline{\eta_{eff}}$	$\overline{r_{intr}}$
simple-synthetic	0.385	1.02×10^{10}	0.034	4.17×10^{9}
fold1000	0.263	6.08×10^{9}	0.029	1.88×10^{9}
fold2000	0.263	6.08×10^{9}	0.029	1.87×10^{9}

Tabela 3. Resultados para o modelo de tempo trânsito T_{OCT} para cada dado sísmico.

3.2. Métrica de desempenho

A ferramenta *Cloud-PITS* avalia o desempenho de cada nó dinamicamente, podendo escolher desligá-lo caso ele não esteja apresentando um desempenho satisfatório. Para tal, o sistema necessita de uma métrica precisa capaz de avaliar o desempenho instântaneo de cada máquina. A taxa temporal das interpolações efetuadas para o cálculo da *semblance* pode oferecer uma boa maneira de computar tal desempenho, já que representa uma medida real de quantos traços a GPU está efetivamente processando por segundo. Sendo assim, definimos formalmente a taxa $r_{intr} = \frac{N^m_{intr}}{\Delta t_m}$, em função do número de interpolações N^m_{intr} e do tempo necessário Δt_m para o cálculo de todo $(t_0, m_0 = m, h_0)$.

Particularmente para o modelo *Offset Continuation Trajectory*, nem todos os traços do conjunto de entrada podem ser utilizados no cálculo da *semblance* para um ponto (t_0, m_0, h_0). Deste modo, introduzimos a métrica $\eta_{eff} = 1 - \frac{N_{out}}{N}$ para medir a porcentagem de traços que foram descartados durante o cálculo em função dos parâmetros N e N_{out}, que representam, respectivamente, a quantia total de traços copiados para o dispositivo e quantos dentres estes não foram utilizados.

3.3. Resultados preliminares

As Tabelas 1, 2 e 3 sumarizam os resultados obtidos conforme configuração apresentada na subseção anterior. Para cada uma delas, dois tipos de execuções são apresentados: uma que utiliza a técnica de seleção de traços apresentada na seção 2, representada por $\mathbb{S}_{CMP}$, $\mathbb{S}_{ZOCRS}$ e $\mathbb{S}_{OCT}$, e a outra que considera todo o conjunto de traços lidos na entrada, $\mathbb{S}_{all}$, no cálculo de cada $P(m_0, h_0, t_0)$. Além disso, $\overline{\eta_{eff}}$ e $\overline{r_{intr}}$ simbolizam, respectivamente, a eficiência e taxa de interpolação médias obtidas para cada execução.

Tanto no modelo *Common Mid Point* quanto no *Zero Offset Common Reflection Surface*, verifica-se que a eficiência média é máxima no caso em que os traços são selecionados. Isso ocorre porque, em ambos, a seleção de um traço dado pela tupla (m_0, h_0) depende exclusivamente de m_0. No caso em que todos os traços são empregados, por outro lado, verifica-se que tais valores são praticamente nulos. Observa-se também que quanto maior a eficiência, maior é a taxa de interpolações por segundo: na tabela 1, a taxa relativa a $\mathbb{S}_{all}$ é praticamente 20 vezes inferior àquela em que a seleção foi realizada, enquanto que, para a tabela 2, a mesma razão se aproxima de 4 vezes.

Por fim, a eficiência obtida para o modelo *Offset Continuation Trajectory* é inferior às

anteriores, porém, ainda assim, é bem superior ao caso em que a seleção não é efetuada. Além de reduzir a quantidade de dados a ser transferida à GPU, os ganhos na taxa de interpolações por segundo foi na ordem de 3 vezes.

4. Conclusões

Neste trabalho, implementamos uma solução de procura dos parâmetros $P(m_0, h_0, t_0)$ totalmente paralelizável e compatível com placas de vídeo suportando *CUDA* e/ou *OpenCL* para três modelos de tempo de trânsito distintos, sendo eles o *Common Mid Point*, *Zero Offset Common Reflection Surface* e *Offset Continuation Trajectory*. Adicionalmente, a solução provida também é compatível com o *frameworks Cloud-PITS*, capaz de executar o cálculo de *semblance* em diversos nós de uma rede computacional. Além disso, o método de obtenção de empilhamentos livres de estiramento [Faccipieri et al. 2019] também foi implementado e faz parte da solução apresentada.

Por fim, um algoritmo de seleção de traços foi desenvolvido. Segundo os testes executados com três dados sísmicos distintos, tal técnica é capaz de aumentar a eficiência do uso dos recursos de uma GPU, com o consequente aumento da taxa de interpolações por segundo. Para o modelo *CMP*, o aumento foi de aproximadamente 20 vezes, enquanto que para o *ZOCRS* e *OCT*, houve uma melhora de, respectivamente, 4 e 3 vezes.

Agradecimentos

Os autores agradecem à FAPESP (CCES 13/08293-7), ao CNPq (140653/2017-1) e à Petrobras pela ajuda financeira e ao Laboratório Multidisciplinar de Alto Desempenho pelo suporte computacional.

Referências

Barros, T., Ferrari, R., Krummenauer, R., and Lopes, R. (2015). Differential evolution-based optimization procedure for automatic estimation of the common-reflection surface traveltime parameters. *Geophysics*, 80(6):WD189–WD200.

Borin, E., Rodrigues, I. L., Novo, A. T., Sacramento, J. D., Breternitz, M., and Tygel, M. (2015). Efficient and fault tolerant computation of partially idempotent tasks. In *14th International Congress of the Brazilian Geophysical Society & EXPOGEF, Rio de Janeiro, Brazil, 3-6 August 2015*, pages 367–372. Brazilian Geophysical Society.

Coimbra, T. A., Novais, A., and Schleicher, J. (2012). Offset continuation (oco) ray tracing using oco trajectories. *Studia Geophysica et Geodaetica*, 56(1):65–82.

da Silva, H. C. (2017). Aceleração de métodos de processamento sísmico com opencl. Master's thesis, Universidade de Campinas, Campinas.

Faccipieri, J. H. (2012). Separação e processamento de difrações em dados geofísicos de reflexão. Master's thesis, Universidade de Campinas, Campinas.

Faccipieri, J. H., Coimbra, T. A., and Bloot, R. (2019). Stretch-free generalized normal moveout correction. *Geophysical Prospecting*, 67(1):52–68.

Gimenes, T. L., Pisani, F., and Borin, E. (2018). Evaluating the performance and cost of accelerating seismic processing with cuda, opencl, openacc, and openmp. In *2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 399–408. IEEE.

Mendonça, A., Silveira, D., Alvarenga, D., and Barreira, G. (2019). Cessão onerosa: governo arrecada r$ 69,96 bilhões com megaleilão do pré-sal. *G1 Globo*.