Melhorando o Desempenho e a Eficiência Energética do Método Fletcher para Simulação de Extração de Petróleo*

Matheus S. Serpa[1], Pablo J. Pavan[1], Jairo Panetta[2], Alexandre Carissimi[1], Philippe O. A. Navaux[1]

[1] Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970, Porto Alegre – RS – Brasil

{msserpa, pjpavan, asc, navaux}@inf.ufrgs.br

[2]Divisão de Ciência da Computação – Instituto Tecnológico de Aeronáutica (ITA)
São José dos Campos – SP – Brasil

jairo.panetta@gmail.com

Resumo. *O Método Fletcher é a base das ferramentas de simulação de extração de Petróleo utilizadas pela indústria. Para realizar tais simulações, arquiteturas paralelas são utilizadas, fornecendo resultados mais rápidos e com maior precisão. Entretanto, para atingir alto desempenho, vários desafios devem ser levados em consideração. Neste artigo, melhoramos o desempenho e a eficiência energética do Método Fletcher em até 73,6% e 42,6%, respectivamente.*

1. Introdução

A geofísica de exploração é fundamental na procura de recursos energéticos, como Petróleo e Gás, mas, altos custos de perfuração, com menos de 50% de precisão por broca, limitam seu uso [Lukawski et al. 2014]. Assim, a indústria de Petróleo e Gás conta com *softwares* de simulação como uma forma economicamente viável de reduzir custos. Vários desafios devem ser levados em conta para se atingir alto desempenho nas arquiteturas que executam esses *softwares*. Um dos aspectos mais importantes é o comportamento do subsistema de memória, já que o acesso à memória possui um papel fundamental no desempenho dessa classe de aplicações [Serpa et al. 2019]. O consumo de energia e a eficiência energética também são pontos a serem considerados.

Neste artigo, trabalhamos com o Método *Fletcher*, uma aplicação de propagação de ondas utilizada por empresas de Petróleo e Gás. Nosso objetivo é otimizar a aplicação matematicamente explorando o seu paralelismo intrínseco.

2. Versão Otimizada e Projeto Experimental

O tempo de execução das derivadas cruzados foi comparado com o da aplicação, mostrando que, o cálculo das derivadas cruzadas demandava 76% do tempo da versão original. Nesse sentido, buscamos reduzir a quantidade de operações para calcular as derivadas cruzadas. A alteração realizada consiste em calcular a derivada cruzada em xy como primeira derivada em y da primeira derivada em x. Isso não reduz a quantidade de operações

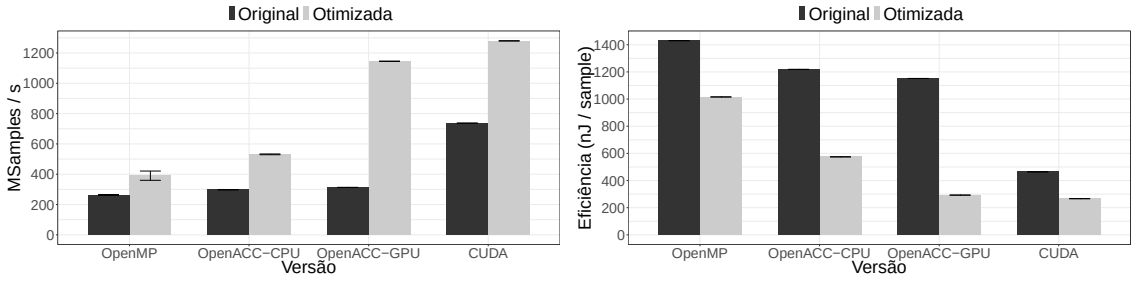
*Este trabalho foi parcialmente financiado pelo projeto Petrobras 2016/00133-9, pelo projeto Petrobras 5900.0111175.19.9 e pelo projeto "GREEN-CLOUD: Computação em Cloud com Computação Sustentável" (#16/2551-0000 488-9), da FAPERGS e do CNPq, programa PRONEX 12/2014.

no cálculo da derivada cruzada em um ponto, mas reduz a quantidade de operações no cálculo da derivada cruzada em y consecutivos, por reutilizar derivadas previamente calculadas em x. Como a mesma redução ocorre nas demais derivadas cruzadas, isso foi aplicado para o cálculo de todas as derivadas cruzadas.

Os experimentos foram realizados em dois ambientes. A *Broadwell* possui dois processadores *Intel Xeon E5-2699 v4* de 22 núcleos, permitindo a execução de até 88 *threads*. A *Pascal* é uma *GPU NVIDIA P100* com 3584 *CUDA Cores*. As codificações *OpenMP*, *OpenACC* e *CUDA* foram executadas 30 vezes sendo que os gráficos mostram os valores médios de tempo de execução e os intervalos de confiança de 95% segundo a distribuição *t de Student*.

3. Resultados Preliminares e Conclusão

Os resultados apresentam o desempenho em *samples* por segundo e a eficiência energética em *Joules* por *sample*. A métrica *samples* por segundo é dada pela divisão do número de pontos da grade calculados (passados via parâmetro de execução) pelo tempo de execução da aplicação. O consumo de energia é obtido via IPMI (*Intelligent Platform Management Interface*). Por fim, a eficiência energética é obtida dividindo a energia pela quantidade de *samples*, ou seja, a quantidade de energia consumida para calcular um *sample*.



(a) Desempenho em *samples* por segundo.

(b) Eficiência Energética em nano *joule* por *sample*.

Figura 1. Desempenho e Eficiência Energética das versões *original* e *otimizada*.

Os resultados mostraram que a versão com maior desempenho foi a versão *CUDA*, seguida da versão *OpenACC-GPU*. A diferença de desempenho entre essas versões foi de 10%. Isso mostra que as otimizações matemáticas podem melhorar o desempenho das aplicações, podendo ser utilizadas em conjunto com otimizações de arquitetura. Após, mostramos que as versões *CUDA* e *OpenACC-GPU* tiverem a melhor eficiência energética, 266 e 292 nano *joules* por *sample*.

Referências

Lukawski, M. Z., Anderson, B. J., Augustine, C., Capuano Jr, L. E., Beckers, K. F., Livesay, B., and Tester, J. W. (2014). Cost Analysis of Oil, Gas, and Geothermal Well Drilling. *Journal of Petroleum Science and Engineering*, 118:1–14.

Serpa, M. S., Cruz, E. H., Diener, M., Krause, A. M., Navaux, P. O. A., Panetta, J., Farrés, A., Rosas, C., and Hanzich, M. (2019). Optimization strategies for geophysics models on manycore systems. *The International Journal of High Performance Computing Applications*, 33(3):473–486.