Análise de Desempenho do Traçamento de Raios Utilizando Nuvem Computacional e o Modelo SPITS

Marcelo M. Vilela Filho[1], **Gustavo B. D. Ignácio**[1], **Nicholas T. Okita**[1], **Edson Borin**[1]

[1]Centro de Estudos de Petróleo (CEPETRO)
Universidade Estadual de Campinas (UNICAMP)
Caixa Postal 6052 – 13.083-970 – Campinas – SP – Brazil

{marcelo,nicholas.okita,ignaciogbd}@ggaunicamp.com, edson@ic.unicamp.br

Resumo. *Este trabalho tem como objetivo investigar o desempenho do traçamento de raios empregado no imageamento sísmico utilizando a nuvem computacional, visando maximizar a escalabilidade do mesmo e minimizar o tempo de execução. Os resultados obtidos mostram que (i) balancear a carga de trabalho alterando a quantidade de raios traçados por nó apresenta redução nos tempos de execução e (ii) a escalabilidade depende da quantidade total de raios.*

1. Introdução

Computação em nuvem é um modelo que disponibiliza uma rede de acesso compartilhado a uma série de recursos computacionais configuráveis que podem ser rapidamente acionados ou desligados de maneira global, sob demanda e com o mínimo de esforço de gerenciamento ou interação com o provedor do serviço[Mell and Grance 2011].

Com a disseminação de seu uso para realizar computação de alto desempenho, diversas áreas de pesquisa têm aproveitado tal recurso para a execução de programas complexos e que demandam alta capacidade computacional. Com isso, o emprego de técnicas de implementação desses programas, como por exemplo, o uso interfaces para programação paralela, que exploram esses recursos, tem alcançado maior importância, visando menores tempos de execução e uso eficiente dos recursos. Uma análise importante para definir a viabilidade do uso de serviços que oferecem *clusters* para a execução de programas é a escalabilidade de seu desempenho, ou seja, a diminuição no tempo de execução do programa com o emprego de mais poder computacional, seja adicionando mais máquinas, seja colocando máquinas com processadores de maior capacidade.

Este trabalho tem como foco a análise de desempenho na nuvem computacional da Amazon (*Amazon Web Services, ou AWS*) de um programa que realiza o traçamento de raios para posterior uso no processamento sísmico. O traçamento de raios é uma etapa importante na área sísmica para extrair informações das estruturas internas da subsuperfície da terra, seja visando a identificação de reservatórios de fluidos seja para a análise da composição geológica da região estudada. Para isso, são resolvidas diversas equações diferenciais, onde métodos numéricos são usualmente empregados para prover as soluções desejadas. Tais soluções determinam trajetórias preferenciais para a propagação da energia, denominadas raios. [Alves and Biloti 2011].

2. Materiais e métodos

Para os testes realizados, utilizamos os serviços de nuvem computacional *Amazon Web Services*, ou *AWS*, da Amazon, especificamente as instâncias do segmento EC2 e todas elas do tipo c5n.large. Cada instância possui 2 vCores (*threads* virtuais) baseados no processador Intel Xeon *Platinum* 8124M e 5,25 GB de memória RAM. Além disso as instâncias eram alocadas no mesmo *placement group* do tipo *cluster*, fazendo com que fiquem no mesmo agrupamento lógico, diminuindo latência de conexão entre elas.

O programa de traçamento de raios foi implementado utilizando o modelo de programação *Scalable Partially Idempotent Task System (SPITS)*, utilizando a plataforma PY-PITS [Borin et al. 2016] para execução. O SPITS é um modelo de computação distribuída baseado no paradigma *Bag-of-Tasks (BoT)* que facilita a execução e implementação de algoritmos paralelos ou distribuídos, abstraindo do programador detalhes da implementação. A ferramenta utilizada para compilação foi o compilador g++ na versão 7.3.1, utilizando a flag de otimização "O3". A implementação conta com o uso da API *OpenMP* na versão 4.5 para paralelização dentro de uma mesma instância. Para a execução do PY-PITS foi utilizado o *runtime* Python na versão 3.

Na plataforma PY-SPITS, o *job manager* divide o trabalho, que podemos definir como a computação total a ser realizada, em tarefas menores, tais tarefas são a unidade mínima de computação, que uma vez finalizadas pelos *workers*, são enviadas ao *commiter*, associando todas as tarefas computadas. Assim, os *workers* processam diversas tarefas, a fim de computar o trabalho total.

A bateria de testes envolveu a modificação de três variáveis no programa. Sendo elas: (i) o número de instâncias que irão processar o programa, variando de 1 a 5; (ii) a quantidade de raios que define uma tarefa, ou seja, quantos raios cada nó irá traçar por tarefa executada, variando entre 10 a 2000 raios, e por fim, (iii) o tamanho total do trabalho, a quantidade total de raios a serem traçados variando entre valores de 10000 a 200000 raios.

Para cada situação, foram feitos 5 testes, medindo-se o tempo de execução do traçamento de raios, e calculando-se a média e o desvio padrão destes tempos para validação estatística dos dados, no entanto, para não dificultar a visualização dos gráficos, omitimos o desvio padrão, mas este possui o variação menor que 15%. Além disso, utilizamos uma máquina dedicada para a função de *Job Manager* e *Committer*, entidades do PY-PITS.

3. Resultados e Discussões

Dividiremos os resultados e as discussões neste trabalho em duas partes, uma referente aos testes em que variamos o número de raios que cada instância irá processar por tarefa executada, ou seja, o tamanho de cada tarefa, e outra parte referente aos testes variando a quantidade total de raios.

3.1. Balanceamento da carga de trabalho

A Figura 1 indica os diversos tempos de execução para dois volumes distintos de trabalho: 100 mil raios e 10 mil raios.O eixo horizontal indica o número de instâncias utilizadas no experimento enquanto as diferentes curvas indicam a quantidade de raios por tarefa.

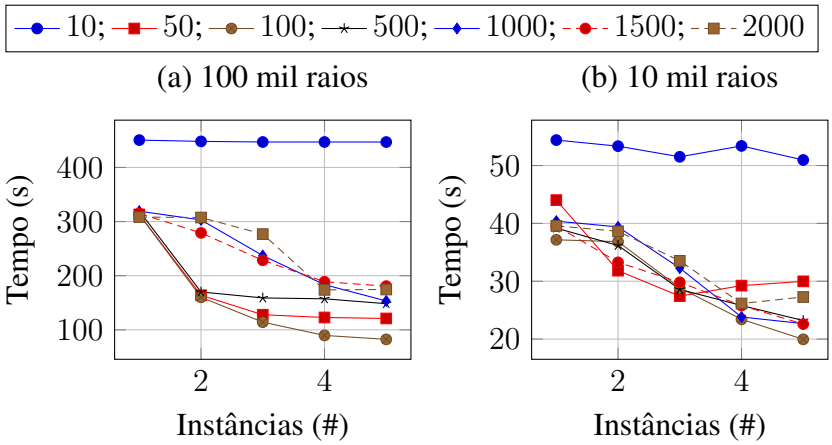

Figura 1. Comparação entre os tempos de execução nas diferentes divisões de trabalho.

A Figura 2, indica o ganho de desempenho obtido para cada número de instâncias, mantendo fixo um total de cem mil raios e variando o tamanho das tarefas, ou seja,a quantidade de raios a serem traçados por tarefa.

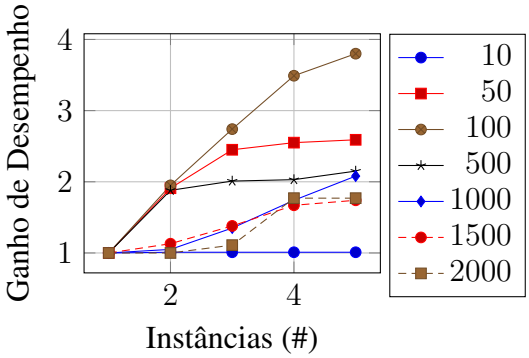

Figura 2. Comparação da escalabilidade nas diferentes divisões de trabalho.

A configuração de cem mil foi escolhida uma vez que como dez mil raios possui um tempo de execução curto, dificultando a visualização dos resultados. Comparando as curvas de desempenho da Figura 2 para cada situação, verificamos que dividir o trabalho total em tarefas de tamanhos próximos a 100 raios possuem um desempenho superior às outras situações, sugerindo a existência de um número ótimo para o balanceamento das cargas de processamento. Por exemplo, para 4 instâncias, no caso de 100 mil raios, temos um tempo de 89,70s agrupando em 100 raios por vez, enquanto para 50 e 500 raios por vez, que são as configurações com desempenho mais próximo a 100, temos 122,86s e 157,35s respectivamente, representando um desempenho 36,9% e 75,4% superior com um agrupamento de 100 raios. Além disso, agrupamentos de 100 raios tem escalabilidade maior do que outros valores, conseguindo escalabilidade de 3,79 vezes com 5 máquinas.

Os testes com agrupamentos de 10 raios por vez apresentaram os piores desempenhos. A hipótese é que, como estamos trabalhando com tarefas muito pequenas, e assim com um tempo de computação curto, tal tempo fica próximo ao tempo de distribuição dos trabalhos somado à latência de transmissão das tarefas para os *workers* e dos resultados para o *committer*, assim, essas latências passam a consumir boa parte do tempo total, impactando o desempenho e tornando-se um gargalo. Tal circunstância foi corrobo-

rada através da análise do *log* de execução, que possui os tempos de início e término da computação de cada tarefa por cada *worker*.

3.2. Escalabilidade em relação à quantidade total de raios

A Figura 3 mostra gráficos da escalabilidade (Ganho de Desempenho x número de instâncias) em cenários com uma quantidade de raios total maior. Como o melhor agrupamento de raios que encontramos foi 100 raios por tarefa, utilizamos essa configuração.

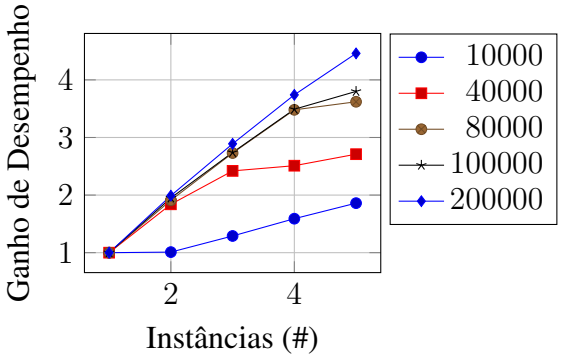

Figura 3. Comparação entre a escalabilidade em diferentes quantidades totais de raios.

Comparando as inclinações das curvas do gráfico, podemos perceber um Ganho de Desempenho maior para quantidades maiores de raios. Nesse contexto, maiores quantidades de raios possuem uma melhor escalabilidade, por exemplo, utilizar 5 máquinas para realizar o traçamento de 200 mil raios possui um Ganho de Desempenho de cerca de 4,5 vezes, enquanto para 10 mil raios, a escalabilidade é de somente 1,8 vezes.

4. Conclusão

O trabalho avaliou o uso do modelo SPITS no programa de traçamento de raios utilizando a nuvem computacional AWS. Os resultados mostram que o traçamento de raios possui boa escalabilidade. No entanto, para atingir tal resultado, é necessário uma avaliação da quantidade de raios traçados por vez para cada instância, ajustando assim o programa para maior eficiência. Deve-se atentar também para a quantidade total de raios traçados, pois esta pode limitar a escalabilidade do problema.

5. Agradecimentos

Os autores gostariam de agradecer à Petrobras, ao CNPq (313012/2017-2) e à Fapesp (2013/08293-7 e 2016/19115-0) pelo apoio financeiro.

Referências

Alves, P. and Biloti, R. (2011). Traçamento de raios em gpgpus. In *12th International Congress of the Brazilian Geophysical Society & EXPOGEF*, pages 1670–1672.

Borin, E., Benedicto, C., L. Rodrigues, I., Pisani, F., Tygel, M., and Jr, M. (2016). Py-pits: A scalable python runtime system for the computation of partially idempotent tasks. In *2016 International Symposium on Computer Architecture and High Performance Computing Workshops (SBAC-PADW)*, pages 7–12.

Mell, P. and Grance, T. (2011). The nist definition of cloud computing. *Communications of the ACM*, 53.