EP3: Cálculo Distribuído do Conjunto de Mandelbrot

Felipe Brigalante 8941280
Mateus Rocha 8941255

MAC 5742-0219 Introdução à Programação Concorrente, Paralela e Distribuída

1. Apresentação e Análise de Medições para o Programa Sequencial

1.1. Detalhes de Implementação

Utilizamos o mesmo código do primeiro execício-programa, para as medições utilizamos a implementação sem alocação de memória e sem I/O.

1.2. Resultados

Utilizamos a imagem de tamanho 8192 e obtivemos os resultados apresentados na Figura 1. O desvio padrão não é indicado pois este é muito pequeno, a variação dos resultados do *perf* foi de no máximo $0,16\%$.

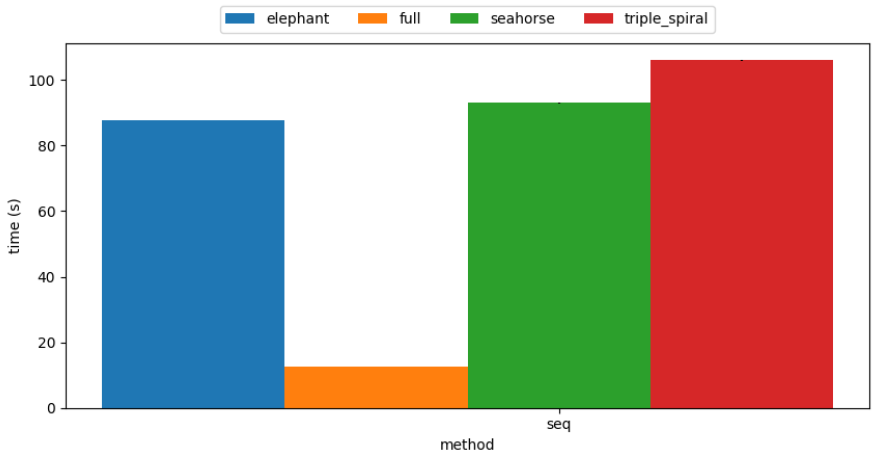

Figura 1: Sequencial sem alocação e I/O

Assim como no EP1, a região "full"é calculada consideravelmente mais rápido que as outras e isso se deve ao fato de que possivelmente os pontos considerados nessa região precisam de menos iterações para convergir.

2. Apresentação e Análise de Medições para o Programa Distribuído

2.1. Detalhes de Implementação

A implementação do *Open MPI* foi realizada da seguinte maneira:

- Os dados são divididos igualmente entre os processos disponíveis, sendo que cada processo decide qual parte da imagem é sua responsabilidade a partir do seu *task id*.
- Depois do processamento da sua parte, cada processo envia uma mensagem pelo *MPI* para a *MASTER* indicando as iterações necessárias para cada ponto.
- A *MASTER*, depois de processar sua parte, recebe as mensagens dos outros processos e adiciona os resultados obtidos nos respectivos pontos da imagem.
- Depois que todas as mensagens são recebidas e processadas, a *MASTER* gera a imagem final.

Não foi utilizado *OpenMP* pois como cada máquina recebe o número de processos igual à quantidade de *cores* disponíveis, a paralelização por *OpenMP* só dificultaria o escalonamento feito pelo sistema operacional e não traria nenhum ganho de desempenho.

2.2. Resultados

Para a realização do experimento foi utilizada uma imagem de tamanho 8192. Na legenda, mpi_{#x} representa a execução da implementação com Open MPI com #x instâncias.

A partir da Figura 2 percebe-se que, comparando as execuções com 1, 2 e 4 instâncias, quanto mais máquinas maior o tempo de execução. Isso acontece pois quanto mais instâncias, mais tempo é gasto com comunicação e como o total de *cores* disponibilizados é o mesmo não há ganho de desempenho.

Em contrapartida, a execução com 8 instâncias apresentou tempos equiparáveis ao de uma única instância. Possivelmente, isso acontece pois como cada máquina é responsável por um único processo, o sistema operacional não precisa lidar com o escalonamento o que acaba otimizando a execução.

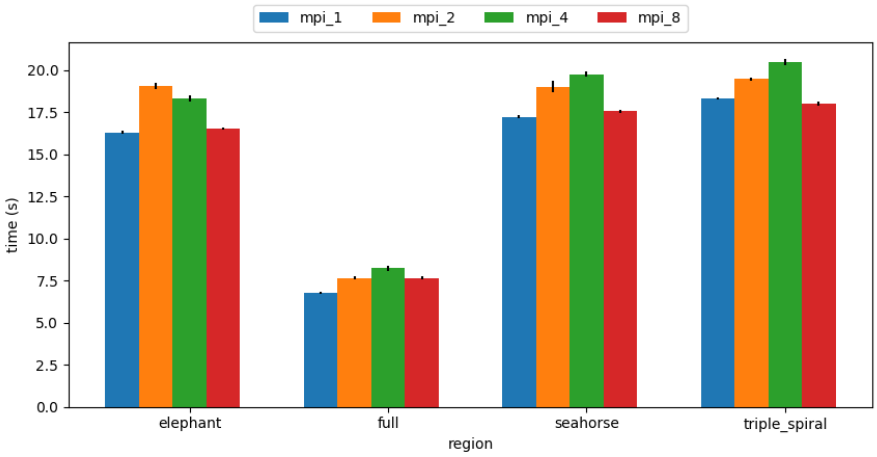

Figura 2: Tempo da implementação distribuída

3. Discussão sobre as Medições

Como indicado na Figura 3, todas execuções de *Open MPI* foram mais rápidas que a sequencial. Em particular, a região *full* tem tempos mais próximos, pois a parte paralelizável do tempo é pequena. Nas demais regiões o ganho é mais significativo, mas não chega a ser 8x mais rápido mesmo utilizando-se 8 *cores*, pois utiliza-se um tempo extra para preparar o *MPI* e também nas trocas de mensagem.

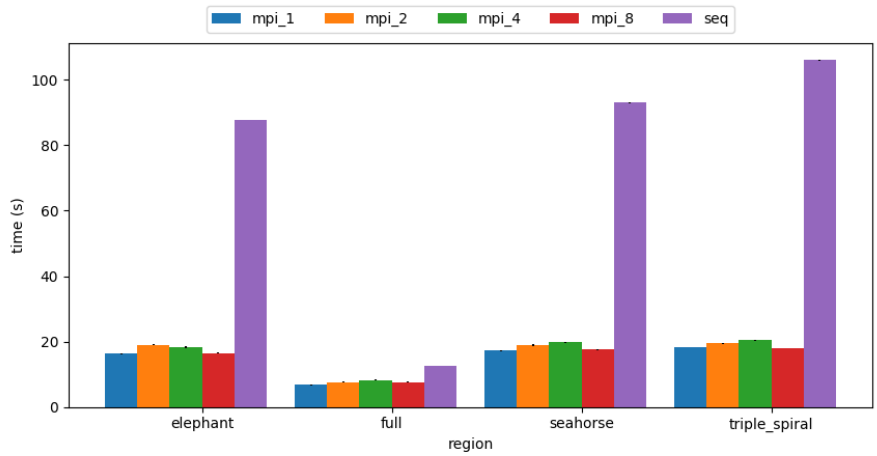

Figura 3: Comparação com todas implementações

Um ponto relevante do *MPI* é que ele disponibiliza uma grande quanti-

dade de *cores*, nesse experimento foram utilizadas no máximo 8 máquinas, porém o programa é facilmente executado com uma quantidade muito maior de máquinas.

Com base nos resultados, conclui-se que a execução com muitas máquinas simples (com apenas um único processador) tem desempenho semelhante ao de com uma única máquina com muitos *cores* e tem desempenho superior a várias máquinas com mais *cores* disponíveis.

4. Links

Todo código feito estará disponível no link abaixo depois do dia de entrega do EP.

`https://github.com/mrocha94/MAC5742-0219-EP3`