Programação Concorrente e Paralela

Calculando um Conjunto de Julia

Aluno(s):
Thales V. S. Lobo de Almeida [**16311010**]

Professor:
Alfredo Goldman Vel Lejbman

2024

1 Resultados dos Experimentos

Nesta seção, apresentamos os resultados dos experimentos realizados para analisar o desempenho do programa em diferentes cenários. Os gráficos gerados a partir dos testes realizados com o MPICH são discutidos a seguir. Os testes foram conduzidos com diferentes números de processos e tamanhos de imagem, abordando tanto a medição de tempo quanto o speedup e a eficiência da versão paralela em comparação com a versão sequencial. O código completo, juntamente com um README explicativo, pode ser encontrado no repositório **https://github.com/Thales-Lobo/julia-set**.

1.1 Tempo de Execução x Resolução

O primeiro gráfico gerado mostra o tempo de execução em função da resolução da imagem, para diferentes números de processos. O gráfico pode ser visualizado na Figura 1. A análise deste gráfico revela que, de forma geral, o comportamento do tempo de execução é linear em relação ao aumento da resolução. No entanto, conforme o número de processos aumenta, o tempo médio de execução diminui significativamente.

É possível perceber que o padrão de linearidade se mantém para todos os números de processos analisados (2, 4, 8, 16), mas a cada incremento no número de processos, o tempo de execução tende a diminuir. Isso sugere que a paralelização está contribuindo de forma efetiva para a redução do tempo de execução, e a utilização de mais processos melhora o desempenho, especialmente em resoluções mais altas.

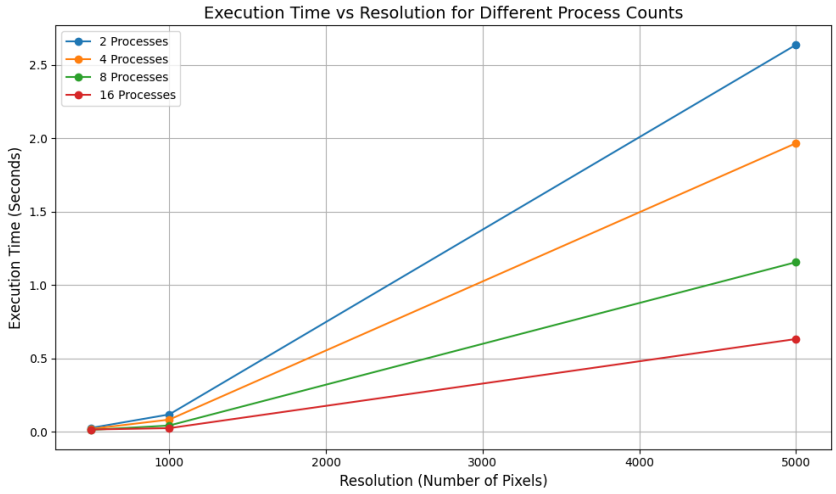

Figura 1: Tempo de Execução em função da Resolução para diferentes números de processos.

1.2 Speedup x Número de Processos

O segundo gráfico gerado apresenta o speedup em função do número de processos para diferentes resoluções. Este gráfico é mostrado na Figura 2. A análise deste gráfico indica que o speedup parece ser quase independente da resolução da imagem, com o valor de speedup crescendo de forma quase linear com o aumento no número de processos. No entanto, para a resolução de 500, observamos uma quebra no comportamento do speedup a partir de 16 processos. Este fenômeno sugere que a utilização de um número excessivo de processos para uma resolução baixa (com menor carga computacional) não resulta em um ganho significativo de desempenho, o que pode ter causado essa queda no speedup.

A partir disso, podemos concluir que o número de processos tem uma influência muito maior no speedup do que a resolução da imagem, especialmente para resoluções maiores, onde o ganho com a paralelização se torna mais evidente.

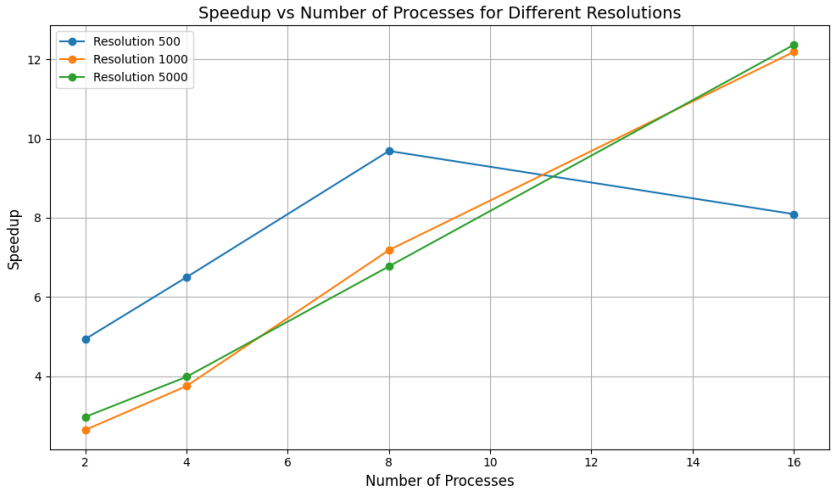

Figura 2: Speedup em função do número de processos para diferentes resoluções.

1.3 Tempo de Execução x Número de Processos (Resolução 4000)

O último gráfico examina o tempo de execução em função do número de processos, com a resolução fixada em 4000 pixels. Esse gráfico pode ser visualizado na Figura 3. A análise desse gráfico mostra uma redução significativa no tempo de execução à medida que o número de processos aumenta, com uma estabilização perceptível a partir de 512 processos.

A partir desse ponto, cada novo aumento no número de processos resulta em uma leve elevação no tempo de execução, sugerindo que o sistema começa a perder eficiência à medida que o número de processos ultrapassa a capacidade ótima de paralelização para a carga de trabalho específica. Isso reflete um fenômeno comum em sistemas paralelos, onde um número excessivo de processos pode gerar sobrecarga de comunicação ou até mesmo saturação do sistema, resultando em um desempenho inferior.

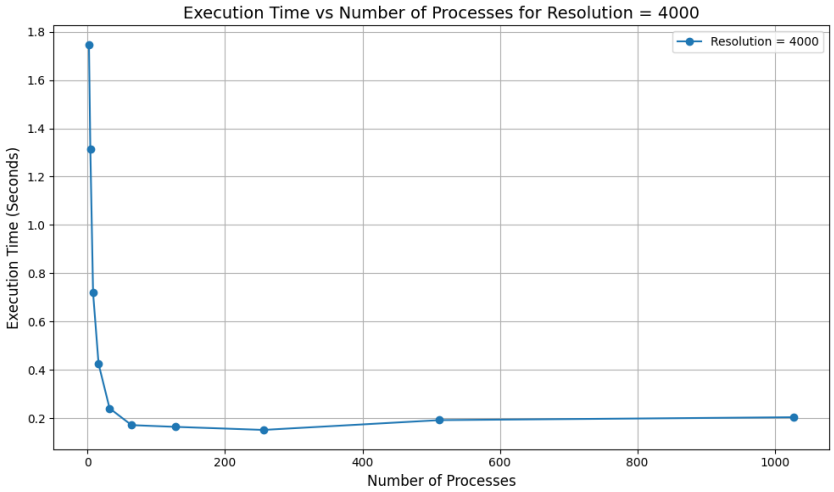

Figura 3: Tempo de Execução em função do número de processos para resolução de 4000 pixels.

2 Experimentos com SimGrid

2.1 Impacto da Heterogeneidade e Homogeneidade dos Nós

Nos primeiros experimentos, foi utilizado o cluster heterogêneo fornecido, que apresenta nós com diferentes capacidades computacionais. O objetivo foi observar como a variação na capacidade dos nós afeta o tempo de execução global. Os resultados indicaram que a heterogeneidade dos nós tem um impacto significativo no desempenho do programa. Quando os nós com maior capacidade eram utilizados para processar as partes mais exigentes do cálculo, o tempo total de execução foi reduzido. No entanto, a presença de nós com menor capacidade aumentou o tempo de execução global, principalmente quando os processos foram distribuídos desproporcionalmente entre os nós. Esse comportamento sugere que, em um ambiente heterogêneo, a distribuição dos processos deve ser feita de forma otimizada para garantir que os nós mais rápidos não fiquem ociosos, enquanto os nós mais lentos não sobrecarreguem o sistema.

Em seguida, o cluster foi modificado para ser homogêneo, ou seja, todos os nós de computação passaram a ter a mesma capacidade. Para isso, o campo *speed* no arquivo `simple_cluster.xml` foi ajustado para garantir que todos os nós possuíssem o mesmo poder de processamento. O objetivo agora era analisar o impacto de uma infraestrutura homogênea no desempenho global. Os testes mostraram que, em um cluster homogêneo, o desempenho foi mais consistente, já que todos os nós contribuíram igualmente para a execução do programa. Comparado ao cenário heterogêneo, o tempo de execução foi significativamente reduzido, uma vez que os processos foram melhor distribuídos entre nós de capacidades similares.

Apesar disso, a paralelização como um todo foi naturalmente menos afetada pela variação de desempenho entre os nós. O desbalanceamento de carga, embora perceptível, não foi tão elevado a ponto de comprometer drasticamente a eficiência global. Esse cenário ilustra que, embora a heterogeneidade afete diretamente o tempo de execução dos processos individuais, a paralelização, de modo geral, foi capaz de mitigar os impactos desse desbalanço, resultando em uma eficiência mais homogênea à medida que o número de processos aumentava. Dessa forma, a homogeneidade dos nós favoreceu a paralelização, resultando em uma melhoria de desempenho, especialmente quando comparado ao ambiente heterogêneo.

2.2 Analise da Latência

Para avaliar o impacto da latência, os testes foram conduzidos simulando três níveis de latência entre os nós: baixa (10 μs), média (100 μs) e alta (500 μs). O campo *link* foi ajustado no arquivo XML para refletir essas condições. O objetivo era observar como diferentes latências entre os nós afetavam o tempo total de execução.

Os resultados indicaram que a latência tem um efeito direto no desempenho do programa, com um aumento significativo no tempo de execução à medida que a latência entre os nós aumentava. Quando a latência era baixa, o tempo de execução foi praticamente o mesmo observado em um ambiente homogêneo, com a paralelização gerando ganhos significativos. No entanto, com a latência média, houve uma leve elevação no tempo total de execução, e com latência alta, o impacto foi ainda mais evidente, especialmente quando o número de processos aumentava. O programa começou a apresentar um desempenho inferior devido à sobrecarga de comunicação entre os nós.

Esses experimentos mostraram que, além da capacidade de processamento dos nós, a latência é um fator crucial para o desempenho de sistemas paralelos em clusters distribuídos. Em cenários com alta latência, a comunicação entre os processos torna-se um gargalo, prejudicando o desempenho global. Portanto, é essencial minimizar a latência ou otimizar a comunicação para sistemas que exigem alto desempenho.