Solução

No exercício B é dito que se aplica o valor de cada RGBA para o pixel, então podemos paralelizar o valor de R, de G, de B e o valor de A:

> Para aplicar o *smooth* nessa imagem colorida, separa-se cada valor da cor primária do pixel e aplica-se o algoritmo. Ou seja, aplica-se o algoritmo de *smooth* para o valor R do pixel, depois para o valor G do pixel, depois para o valor B e, por fim, para o A. Dessa forma, o novo pixel colorido calculado tem novos valores para (R, G, B, A). A Figura B2 mostra esse cálculo apenas para o valor de R de um pixel.

Outra forma de paralelização implementada foi na iteração dos pixels, o que é eficaz para este problema, pois cada pixel pode ser processado de maneira independente. Não há nenhum fator que interfira no cálculo, já que ele é realizado com os valores iniciais da matriz. Isso elimina a necessidade de comunicação entre os threads durante o processamento.

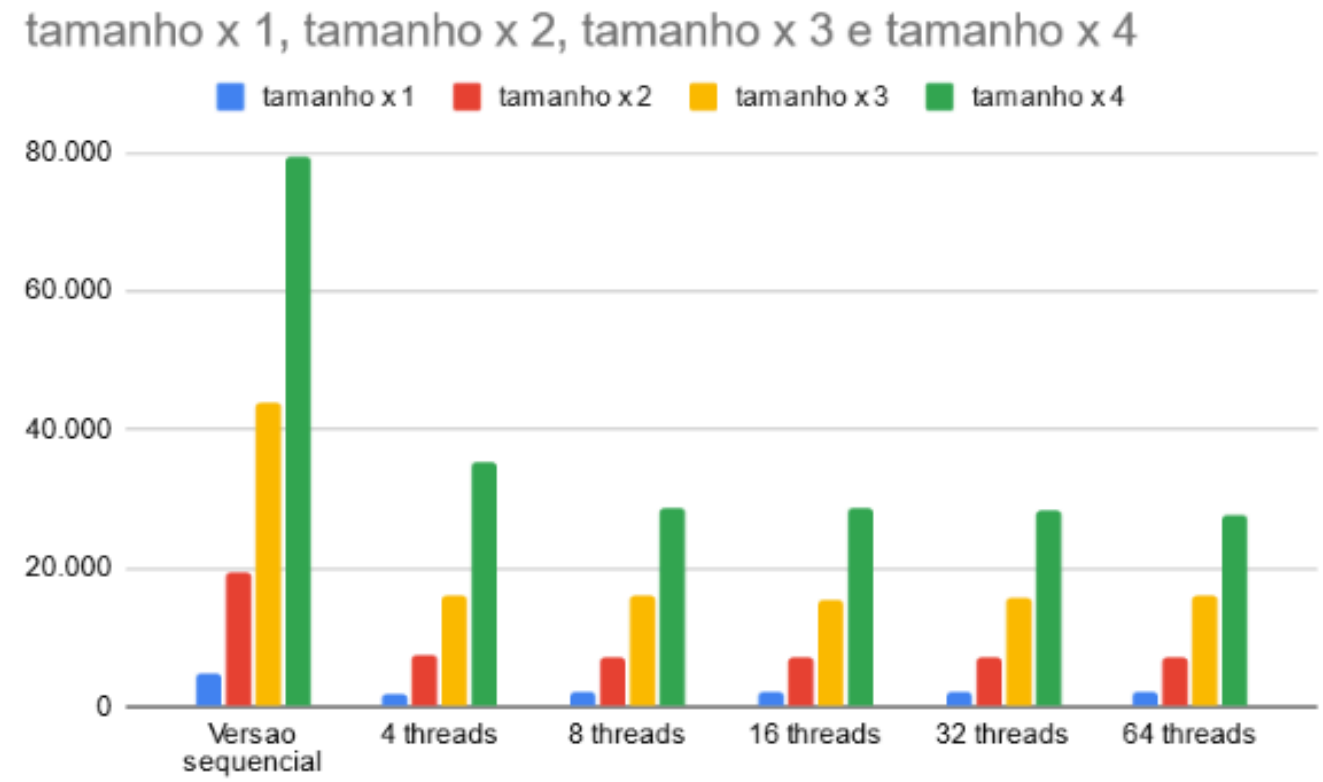
Confirmação de que o algoritmo está funcionando

Foi utilizado o comando diff para ver se havia alguma diferença entre a image.out da versão paralela com a versão sequencial.

```
gacrescencio@Gabriel:~/programacao-paralela/teste$ diff image_sequencial.out image_paralela.out
```

Comparação entre a versão sequencial e a versão paralela com 4,8,16,32 e 64 threads

	tamanho x 1	tamanho x 2	tamanho x 3	tamanho x 4
Versao sequencial	4.687	19.238	44.049	79.374
4 threads	1.813	7.468	16.228	35.148
8 threads	2.120	7.195	15.982	28.583
16 threads	2.269	7.110	15.436	28.588
32 threads	2.259	7.152	15.720	28.305
64 threads	2.228	7.141	15.953	27.594



Tamanho = 7680 x 4320

https://docs.google.com/spreadsheets/d/1DdsqfqoS8GmZpmwG-yv704XK_tmzNwkKBpWn2tsOUhQ/edit#gid=1470436756

Conclusão

O problema de suavização (smooth) envolve a aplicação de um filtro em uma imagem para reduzir o ruído e melhorar a qualidade visual. O filtro é aplicado separadamente para (R, G, B, A), iterando sobre cada pixel da imagem e calculando uma média aritmética de um determinado pixel com os pixels vizinhos. Este processo é repetido para cor separadamente(R,G,B,A), resultando em uma imagem mais suavizada.
Na versão sequencial, o programa iterava por cada pixel da imagem e calculava a média do pixel atual com os pixels vizinhos para suavizar a imagem. Este método, sem a paralelização, estava computacionalmente intensivo, demorando muito para processar grandes imagens.
Para melhorar a eficiência, a implementação foi paralelizada usando OpenMP, distribuindo o trabalho entre múltiplas threads(4,8,16 e 32). A paralelização foi aplicada nos loops que iteram sobre os pixels da imagem e também nos cálculos independentes do (R,G,B,A) que é citado no enunciado.
A versão paralela do programa mostrou uma melhoria significativa no tempo e na performance em comparação com a versão sequencial. A paralelização foi efetiva em reduzir o tempo total de execução, especialmente para imagens maiores, onde conforme era aumentado o número de threads melhor era o desempenho para grandes imagens, é importante destacar que o número máximo de threads usadas foi 64. No entanto, essa melhoria veio com um aumento do uso de CPU e do tempo de sistema, indicando que o paralelismo exigiu mais recursos de gerenciamento.
Além disso, foi observado que o aumento do número de threads proporcionou ganhos significativos de desempenho apenas para imagens maiores. Para imagens menores, teve um ganho significativo com a paralelização, mas o aumento no número de threads não resultou em uma diferença substancial no tempo de execução.