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 paralela e a versão sequencial

Versão Sequencial

- User time (seconds):** 2.72
- System time (seconds):** 0.15
- Percent of CPU this job got:** 99%
- Elapsed (wall clock) time (h:mm:ss or m:ss):** 0:02.87

Versão Paralela

- User time (seconds):** 16.72
- System time (seconds):** 3.86
- Percent of CPU this job got:** 1778%
- Elapsed (wall clock) time (h:mm:ss or m:ss):** 0:01.15

Conclusão

A versão paralela do programa mostrou uma melhoria significativa no tempo de execução (elapsed time), embora com maior uso de CPU e aumento do tempo de sistema. Isso indica que o paralelismo foi efetivo em reduzir o tempo de execução total, mas com um custo maior de gerenciamento de recursos