Segundo Trabalho

Programação Concorrente

Fernando Gorodscy	7152354
Leonardo Rebelo	5897894

Índice

Instruções (Read-me)

1. Compilar os códigos sequencial e paralelo
    a. Abrir o terminal
    b. Entrar na pasta do trabalho
    c. $ g++ -I/usr/include/opencv -I/usr/include/opencv2 -L/usr/include/opencv/lib/ -g -o t2p trab2.cpp -lopencv_core -lopencv_imgproc -lopencv_highgui -lopencv_ml -lopencv_video -lopencv_features2d -lopencv_calib3d -lopencv_objdetect -lopencv_contrib -lopencv_legacy -lopencv_stitching -fopenmp
2. Executar o código
    a. ./t2p <input_file> <output_file>

Algoritmo

Dividimos a imagem em blocos de 4x4. Para cada bloco 4x4 fizemos a média dos valores de cada pixel do bloco dos 3 canais de cores (Vermelho, Verde e Azul). Ou seja, a média dos valores de vermelho dos pixels, do verde e do azul.

Foram feitas 3 implementações do algoritmo para comparação. A primeira sequencial, a segunda paralelizada apenas com OpenMP e a última utilizando OpenMP e OpenMPI.

Utilizamos OpenMP para paralelizar o loop que calcula a média, utilizando todas os cores do nó. Além disso, paralelizamos usando OpenMPI, de forma que cada nó do cluster fique responsável por processar uma parte da imagem. Essa divisão é feita baseando-se na quantidade de nós disponíveis e no tamanho da imagem. Em outras palavras, dividimos a largura e a altura da imagem pelo número de nós - 1 (todos menos o nó mestre).

O nó mestre não participa do processamento da imagem, pois ele fica responsável por gerenciar os nós escravos e reunir os fragmentos de imagem para montar a imagem final.

Outro fato importante de se notar é que cada nó (incluindo o master) faz leitura da imagem. Isso facilita o algorítimo, pois retira a preocupação de envio de parte da imagem do mestre para os escravos, e também retira do mestre a responsabilidade de processar tal divisão. Com isso, diminuímos o processamento do mestre, que geralmente é o gargalo. Porém, adicionamos algum custo de processamento aos nós escravos, que precisam ler a imagem. A pior parte é que pelo fato de a imagem estar localizada no disco rígido do nó mestre, todos os nós precisam que essa imagem seja transferida pela rede do nó mestre para ele. Além de aumentar significativamente o uso da rede, isso aumenta demais o uso do disco rígido do mestre causando uma grande perda de performance.

Hardware Utilizado

Nodes (19 hosts / 72 virtuais) – 18 slaves nodes / 1 master node

Intel® Core™2 Quad Processor Q9400 (6M Cache, 2.66 GHz, 1333 MHz FSB)
8 GB RAM DDR3 Kingston
Motherboard Gigabyte G41-MT-S2P
HD 160GB Seagate Sata II 7200RPM
Fonte ATX 300W Real

Resultado Esperado

Ao pensar em resultado devemos analisar as partes que mudam quando implementamos sequencialmente ou paralelamente. O programa paralelo MPI possui algumas adições, por exemplo, remontar a imagem usando os blocos processados pelos nós escravo, além do overhead causado pela transferência desses dados pela rede, do nó escravo para o mestre.

Sendo assim, o esperado é que ao processar imagens pequenas, o overhead é muito grande, fazendo com que o algoritmo sequencial seja mais eficiente, ou que o paralelo tenha um speedup muito baixo.

Resultados Observados

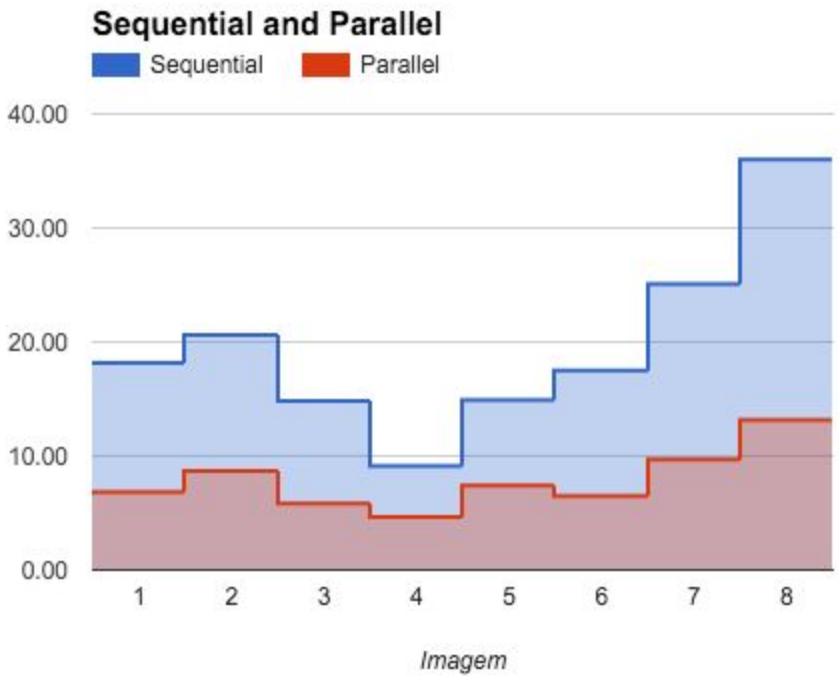
	Imagem 1	**SpeedUp: 2.66**	**Imagem 2**	**SpeedUp: 2.37**
Tipo de algoritmo:	Sequential	Parallel	Sequential	Parallel
Execução # 1	18.18	6.78	20.63	8.90
Execução # 2	18.20	6.76	20.65	8.95
Execução # 3	18.18	7.09	20.60	7.65
Execução # 4	18.21	6.75	20.57	9.32
Execução # 5	18.11	6.80	20.60	8.98
Execução # 6	18.21	7.10	20.60	9.01
Execução # 7	18.18	6.81	20.57	8.99
Execução # 8	18.17	6.73	20.62	7.65
Execução # 9	18.08	6.79	20.61	8.79
Execução # 10	18.17	6.76	20.64	8.80
Média:	**18.17**	**6.84**	**20.61**	**8.70**
Desvio Padrão:	**0.04**	**0.14**	**0.03**	**0.57**

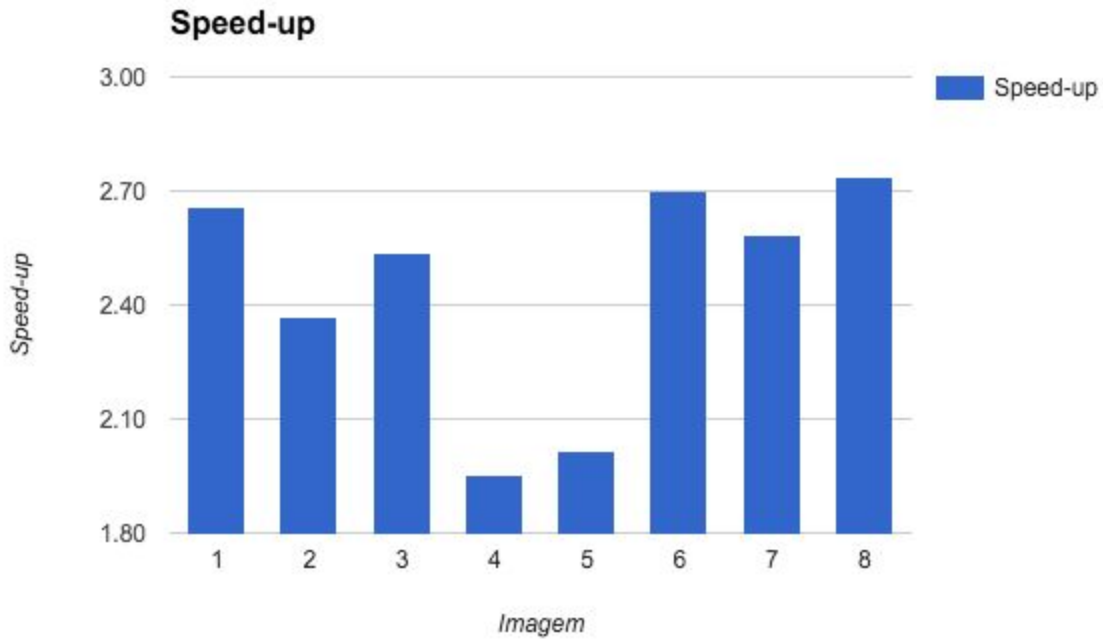
	Imagem 3	**SpeedUp: 2.54**	**Imagem 4**	**SpeedUp: 1.95**
Tipo de algoritmo:	Sequential	Parallel	Sequential	Parallel
Execução # 1	14.84	5.88	9.13	4.62
Execução # 2	14.82	5.86	9.08	4.69
Execução # 3	14.74	5.78	9.13	4.68
Execução # 4	14.85	5.78	9.07	4.85
Execução # 5	14.80	5.76	9.07	4.60
Execução # 6	14.80	5.76	9.09	4.62
Execução # 7	14.83	6.05	9.09	4.60
Execução # 8	14.78	5.78	9.11	4.69
Execução # 9	14.84	5.82	9.12	4.69
Execução # 10	14.82	5.88	9.15	4.62
Média:	**14.81**	**5.83**	**9.10**	**4.67**
Desvio Padrão:	**0.03**	**0.09**	**0.03**	**0.08**

	Imagem 5	**SpeedUp: 2.02**	**Imagem 6**	**SpeedUp: 2.70**

Tipo de algoritmo:	Sequential	Parallel	Sequential	Parallel
Execução # 1	14.94	7.46	17.41	6.48
Execução # 2	14.87	7.77	17.50	6.46
Execução # 3	14.92	7.23	17.53	6.48
Execução # 4	14.88	7.33	17.50	6.40
Execução # 5	14.95	7.42	17.46	6.56
Execução # 6	14.98	7.47	17.43	6.53
Execução # 7	14.88	7.42	17.53	6.40
Execução # 8	14.91	7.33	17.46	6.48
Execução # 9	14.91	7.23	17.55	6.47
Execução # 10	14.93	7.35	17.44	6.51
Média:	**14.92**	**7.40**	**17.48**	**6.48**
Desvio Padrão:	**0.03**	**0.16**	**0.05**	**0.05**

	Imagem 7	**SpeedUp: 2.59**	**Imagem 8**	**SpeedUp: 2.74**
Tipo de algoritimo:	Sequential	Parallel	Sequential	Parallel
Execução # 1	25.11	9.56	35.97	13.49
Execução # 2	25.03	9.50	36.04	12.90
Execução # 3	25.14	9.52	35.96	12.90
Execução # 4	24.99	9.64	36.09	13.11
Execução # 5	25.11	10.52	36.04	13.07
Execução # 6	25.17	10.48	35.88	12.99
Execução # 7	25.17	9.42	35.95	13.07
Execução # 8	25.01	9.31	35.97	13.12
Execução # 9	25.04	9.51	36.14	13.45
Execução # 10	25.00	9.54	36.08	13.44
Média:	**25.08**	**9.70**	**36.01**	**13.15**
Desvio Padrão:	**0.07**	**0.43**	**0.08**	**0.22**


Sequential and Parallel
Sequential
Parallel
40.00
30.00
20.00
10.00
0.00
1
2
3
4
5
6
7
8
Imagem


Speed-up
Speed-up
3.00
2.70
2.40
2.10
1.80
Speed-up
1
2
3
4
5
6
7
8
Imagem

Analise dos Resultados

Pudemos observar que os resultados foram razoavelmente diferente do esperado. A velocidade do algoritmo paralelo foi sempre melhor do que o sequencial, mesmo para imagens menores. Provavelmente porque não escolhemos imagens suficientemente pequenas, ou pelo fato de que a vantagem do processamento paralelo supera muito o overhead causado pela rede.

Possíveis melhorias

Uma melhoria evidente seria evitar que cada nó lesse a imagem do disco. Com isso iriamos diminuir o uso do disco em <num_nodes> vezes. Apesar de aumentar o uso da rede, o uso do disco é muito pior.

Outra melhoria seria não ler (ou transferir) toda a imagem nos nós. Seria muito mais eficiente ler apenas a parte da imagem que o nó processaria. Porém, isso aumentaria consideravelmente o trabalho de desenvolvimento.

Conclusão

O algoritmo paralelo foi claramente mais eficiente que o sequencial, chegando a um ganho de mais de 2.5 vezes o sequencial. Em troca de tal ganho no desempenho, o algoritmo paralelo é mais complexo de ser implementado. Para que haja sucesso na implementação de um algoritmo paralelo, é necessário que se identifique corretamente os pontos paralelizáveis de um algoritmo sequencial e que se calcule corretamente o ganho em relação ao *overhead* causado pelo uso da rede para transferência de dados entre nós e outros mecanismos necessários para a correta implementação paralela.