Trabalho Prático
Programação Paralela

Contagem da quantidade de estrelas em imagem astronômica utilizando o framework MPI

Gustavo Alves Miguel
Luis Gustavo de Souza Carvalho
Ciência da Computação
ICEX – Departamento de Ciências Exatas

Alfenas/2018

Objetivo

O presente trabalho, objetivou-se produzir um sistema para contagem de estrelas de uma imagem astronômica utilizando o conceito de programação paralela e realizar a análise de comportamento do sistema executando em aglomerados de computadores em termos de tempo de execução, tamanho do bloco de dados e quantidade de processos.

A tarefa a ser realizada é contar a quantidade de estrelas numa imagem astronômica (dividida em 20 arquivos PNG), que será dividida em imagens menores pelo computador master e por ele distribuída para os demais computadores slaves. Cada computador slave deve processar a parte da imagem que receber e devolver ao master a quantidade de estrelas daquela parte da imagem. O computador master deve resumir a quantidade de estrelas de todas as partes.

As tomadas de tempo do relatório devem variar o tamanho das partes de imagens distribuídas e a quantidade de slaves utilizados, e os valores de tempo obtidos devem ser analisados ao final do relatório nos termos de eficiência, redundância, utilização e qualidade do sistema.

Metodologia

Para ler a instância do trabalho utilizamos da ferramenta IrfanView para manipular as imagens para o formato PGM e verificarmos a escala de cinza das imagens, com isso aplicamos aos blocos de imagens distribuídos pelos processadores o método de binarização, onde pixels com intensidade maior de 127 na escala de cinza são atribuídos o binário 1 e caso for menor ou igual são atribuídos o binário 0. Como as estrelas têm a cor mais clara diferentemente do espaço, este método nos fornece os elementos que são partes de estrelas na imagem.

Após realizar a binarização, aplicamos o algoritmo *Connected-Component Labeling* analisando vizinhança-4 para rotular componentes de uma imagem. Ao encontrar um ponto com o valor 1, o algoritmo verifica o vizinho esquerdo e superior, caso todos os vizinhos tem o

valor 0, é atribuído um novo rótulo para aquela posição e passa para o próximo ponto, caso os seus vizinhos conterem um rótulo, é atribuído para aquele ponto o rótulo do seu vizinho, e tratamos alguns casos particulares como no caso de os vizinhos apresentarem rótulos diferentes.

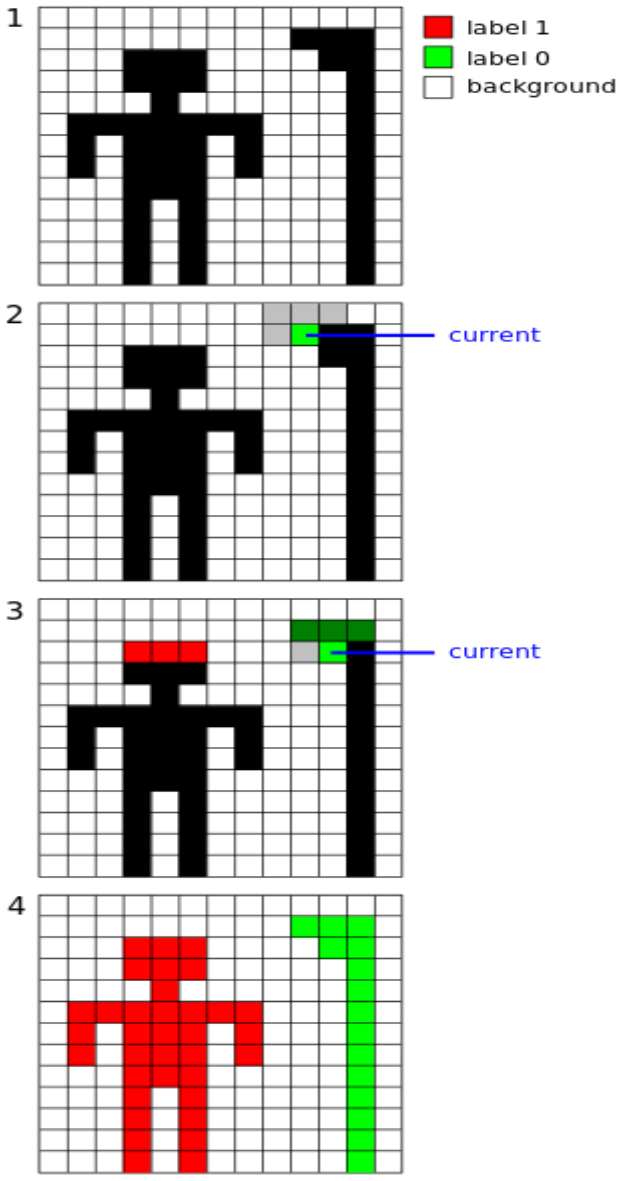

Figura 1: Exemplo de saída gráfica da execução do algoritmo de duas passagens em uma imagem binária. A primeira imagem não é processada, enquanto a última foi recolorida com informações de rótulos (Wikipedia).

Com isso ao final da execução do bloco, temos a quantidade de rótulos utilizados e também eliminando os casos particulares, temos a quantidade de estrelas em determinado bloco.

Assim, podemos somar para cada bloco rotulado a quantidade de estrelas presente, e com isso ao final da imagem, obter a quantidade de estrelas da imagem e ao final da execução temos a soma total de todas as imagens.

Para dividir a execução em processos diferentes, utilizamos o *mpi* onde o processo *root* tem a responsabilidade de enviar para os processos *slaves* os blocos lido do arquivo pgm, assim cada *slave* irá ter responsabilidade de executar o algoritmo de binarização do bloco e rotular os elementos do bloco e retornar a quantidade de estrelas do bloco.

A execução então segue a seguinte ordem cronológica.

1. É efetuada a leitura das imagens pelo *root;*
2. O *root* envia blocos com tamanhos pré definidos aos *slaves*;
3. Cada *slave* executa a binarização e o rotulamento no bloco que lhe foi atribuído;
4. Após feita a contagem, o resultado é somado ao total;
5. No final da execução, o *root* possui o resultado final da contagem de estrelas.

Esse conjunto de operações foi executado com diferentes números de blocos e diferentes números de processos. Alguns dos valores obtidos são mostrados a seguir.

Resultados

Os resultados foram gerados em máquinas com processador i5 e com 8 GB de memória RAM.

Quantidade CPU's	Tamanho Blocos	Tempo de execução	Contagem de estrelas
1 CPU	512	1.158 segundos	104.987.348
1 CPU	256	1.218 segundos	104.989.028
2 CPU's	512	640 segundos	104.987.348
2 CPU's	256	740 segundos	104.989.028
3 CPU's	512	571 segundos	104.987.348
3 CPU's	256	683 segundos	104.989.028

4 CPU's	512	524 segundos	104.987.348
4 CPU's	256	639 segundos	104.989.028

Tabela 1: Resultados encontrados na execução do algoritmo para diferentes quantidades de blocos e diferentes quantidades de computadores.

	1 CPU	2 CPU's	3 CPU's	4 CPU's
Tempo(s)	1.158	640	556	513
Operações(s)	1.893.175	1.896.871	1.896.872	1.896.873
Speedup(s)	1	1,80	2,08	2,25
Eficiência(s)	1	0,90	0,69	0,56
Redundância(s)	1	1,001952	1,001952	1,001953
Utilização(s)	1	0,90	0,69	0,56
Qualidade(s)	1	1,61	1,43	1,26

Tabela 2: Medidas de eficiência para as operações realizadas em blocos de tamanho 512 x 512

	1 CPU	2 CPU's	3 CPU's	4 CPU's
Tempo(s)	1.218	740	683	639
Operações(s)	3.883.923	3.899.067	3.899.068	3.899.069
Speedup(s)	1	1,64	1,78	1,90
Eficiência(s)	1	0,82	0,59	0,47
Redundância(s)	1	1,003899	1,003899	1,003899
Utilização(s)	1	0,82	0,59	0,47
Qualidade(s)	1	1,34	1,05	0,89

Tabela 3: Medidas de eficiência para as operações realizadas em blocos de tamanho 256 x 256

Conclusão

Com os resultados obtidos podemos perceber que, para se obter o melhor resultado, o uso de blocos de tamanho 512 x 512 foi a melhor escolha. Isso ocorre pois, com um tamanho menor de bloco, aumenta-se o número de blocos e, consequentemente, o número de comunicações que devem ser feitas. Vemos também que o há uma melhora pequena entre o uso de 3 e 4 CPU's. Tendo isso, pode-se considerar, caso seja necessário poupar equipamento, o uso de apenas 3 CPU's, porém ainda é viável o uso de quatro tendo em vista que suas medidas de eficiência mantém um valor relativamente alto.

Percebe-se também um pequeno aumento no valor da contagem quando diminuímos o tamanho do bloco. Isso ocorre devido ao aumento do número de bordas. Com mais blocos, temos mais bordas não sendo tratadas.