

MC900/MO644 - Programação Paralela

Laboratório 1

Professor: Guido Araújo
Monitor: Luís Felipe Mattos

Count Sort

O count sort é um algoritmo de ordenação simples, para cada elemento do vetor é feita uma contagem no array do número de elementos menores. Assim, podemos saber em qual posição do array ordenado o elemento ficará. Uma implementação sequencial pode ser vista no exercício 5.3, na página 268 do livro-texto. Porém, uma variação da versão sequencial será fornecida, a fim de evitar divergências de complexidade do algoritmo sequencial.

Como exemplo temos o array [4 2 4 3 1 5].

Passo 1: Para o primeiro elemento, existem 3 elementos menores do que 4 no vetor e um elemento igual, portanto este elemento deve ficar na posição 3 do array ordenado, já que o outro elemento repetido não foi computado ainda. O vetor temporário fica [- - 4 - -]

Passo 2: Para o segundo elemento, temos que somente 1 elemento é menor do que 2, então ficará na posição 1 do vetor de resultado. Ficamos então com seguinte o vetor temporário de resultados [- 2 - 4 - -]

Passo 3: Para o terceiro elemento, por ser um elemento repetido, temos que contar não só quantos elementos são menores, mas também quantas ocorrências do mesmo elemento já foram computadas e somar ao índice no vetor ordenado. Assim, garantimos que os elementos ficarão nas posições corretas no resultado.

Após esta nova contagem, temos o seguinte vetor [- 2 - 4 4 -]

Passo 4: Para o quarto elemento, existem 2 elementos menores do que 3 no vetor, portanto este elemento deve ficar na posição 2 do array ordenado,

o vetor temporário fica [- 2 3 4 4 -]

Passo 5: Para o quinto elemento, não existe nenhum elemento menor do que 1 no vetor, portanto este elemento deve ficar na posição 0 do array ordenado, o vetor temporário fica [1 2 3 4 4 -]

Passo 6: Para o último elemento, existem 5 elementos menores do que 5 no vetor, portanto este elemento deve ficar na posição 5 do array ordenado, o vetor ordenado fica [1 2 3 4 4 5]

Enunciado

Deve-se implementar um count sort paralelo usando os pragmas de Openmp. O programa deve receber como entrada 3 linhas: a primeira que contém um inteiro que representa o número de threads que o programa deve rodar, a segunda linha que também contém um inteiro, o número de elementos do vetor e a terceira linha que contém os elementos do vetor, que são números do tipo float, separados por espaço. O exemplo de entrada a seguir:

```
4
10
10.25  8.31  10.30  8.25  4.05  5.20  12.35  12.50  20.00  2.90
```

Neste exemplo, o programa deve utilizar 4 threads para ordenar o vetor de 10 elementos. O resultado deve ser impresso na saída padrão (stdout) no seguinte formato, a primeira linha contém o vetor ordenado e a segunda linha o tempo de execução do programa em segundos. É recomendado o uso da função *omp_get_wtime()* da biblioteca do Openmp. O exemplo de saída a seguir:

```
2.90  4.05  5.20  8.25  8.31  10.25  10.30  12.35  12.50  20.00
0.000146
```

Chama-se atenção para o vetor de saída, não é preciso verificar se é a última iteração do loop de impressão para adicionar o espaço. **Deve-se**

adicionar o espaço ao final do vetor. Assim a implementação e o entendimento do código ficam mais simples.

Outro detalhe importante é que **os valores do vetor ordenado devem ter 2 casas decimais.**

Testes e Resultado

Quanto aos testes, serão 3 testes abertos e 3 testes fechados de mesmo nível de complexidade. Teremos entradas variando entre 5000, 10000 e 20000 valores float a serem ordenados. A saída do programa será comparada com o vetor ordenado para garantir que a ordem das operações foi feita corretamente em paralelo.

O resultado considerado correto é o vetor ordenado e o programa paralelo deve ter algum speedup em relação ao programa serial. Serão consideradas as duas condições para que o susy considere o programa correto dada a entrada.

O código deve conter comentários das passagens principais, não é preciso comentar cada linha.