Relatório 1.3 Programação concorrente

Caio Eduardo Kikuti Machado - RA103235 - Universidade Estadual de Maringá

***Abstract*—This is a study about the speedup and execution time over the count3 algortithm execution in prallel and sequncial mode.**

I. Introdução

O Conceito de *speedup* na computação é conhecido como a medida relativa da performance de dois sistemas executando o mesmo problema. Normalmente o *speedup* é mensurado em relação a dois parâmetros, o primeiro é a latência que compreende ao tempo de execução de uma tarefa, o segundo é a taxa de transferência de dados. Neste relatório abordaremos o *speedup* comparando o tempo de execução de um algoritmo, *count3*, sob a mesma arquitetura, porém utilizando recursos diferentes.

O experimento baseia-se na execução e cálculo do *speedup* na execução do código *count3*. O primeiro algoritmo foi executado de maneira sequencial ao passo que o segundo algoritmo utilizou recursos de programação paralela sendo executado 4 vezes, uma vez para cada quantidade de *threads* da arquitetura em questão.

Para a realização deste trabalho utilizou-se um computador com a seguinte configuração: processador Intel I5 7200u, 8GB RAM, 2 núcleos físicos e 4 *threads*, linha de memória cahce de 64 bytes.

II. Desenvolvimento

As versões de códigos executadas contemplam o mesmo problema, a contagem de um número *k* em um vetor gerado de maneira aleatório com *p* porcentagem do número *k* contida no mesmo. A variação do algoritmo encontra-se apenas no emprego de recursos de programação paralela, utilizando a biblioteca *pthreads* da linguagem C. Ao observarmos o tempo de execução vemos que em todas as versões de código paralelo houve um ganho em relação ao código sequencial. O tempo total de execução do código sequencial para os 3 tamanhos de vetor escolhidos (500000000, 1000000000, 1870000000) foi maior que o paralelo em todas as ocasiões. Para uma *thread* o código paralelo foi cerca de 12,6% maior já com duas a diferença aumenta para 16,91%, executando o algoritmo com 3 e 4 *threads* o tempo de execução paralelo foi 17,5% e 18,9% menor que o sequencial, respectivamente. Em relação ao *speedup* pode-se notar pela Figura 2 que, de maneira geral as entradas pequenas não demonstraram um ganho tão expressivo, isso deve-se ao fato de que a utilização de computação paralela acaba gerando um gasto a mais de recursos o que acaba por não ser muito proveitoso para problemas com poucos dados. Quando fora utilizado o número de máximo de *threads* alcançou-se o melhor resultado sendo o *speedup* 1,318. Para a maior entrada, ao utilizarmos os 4 fluxos disponíveis aumentou-se o *speedup* em cerca de 3% em relação ao programa que utilizou apenas duas *threads*.

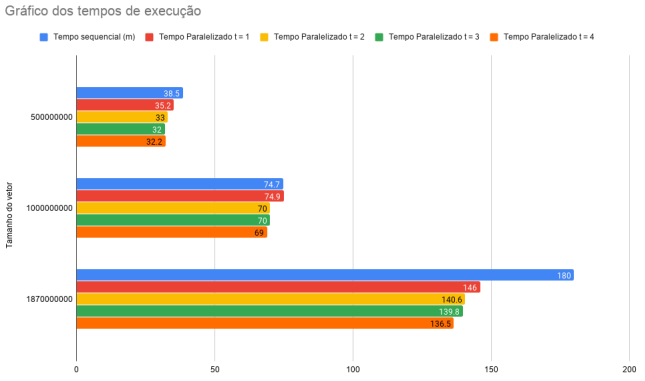

Fig. 1. Gráficos do Tempo de Execução

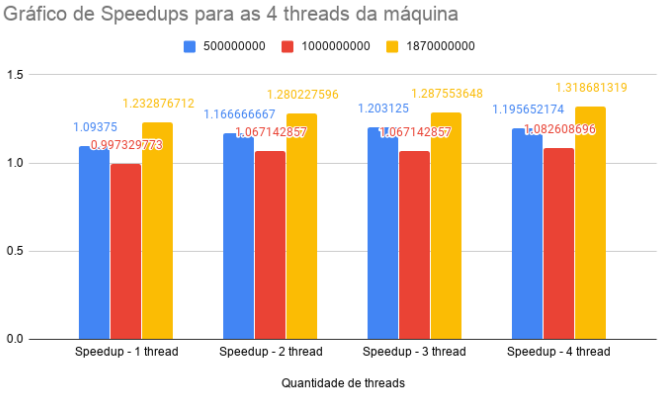

Fig. 2. Gráficos de speedups

III. Conclusão

Pelo experimento podemos concluir que há um ganho evidente quando o código é executado de maneira paralela, e que esse ganho fica mais evidente quando são utilizadas entradas de maior tamanho. Como comentado na Seção 2, a melhoria no tempo de execução aumentou ao passo que houve o aumento do tamanho da entrada, sendo que para a maior entrada conseguiu-se uma melhora de quase 44 segundos no tempo de execução. E portanto essa melhora também refletiu no *speedup* calculado.

References

[1] Faustino, Anderson. Paralelismo usando várias Threads Disponível em: https://moodlep.uem.br/pluginfile.php/159503/mod_resource/content/1/ParalelismoUsandoV%C3%A1riosThreads.pdf em: 29 ago. 2020

Caio Kikuti Aluno do quarto ano do Curso de Ciência da Computação UEM