O Paralelismo no problema contar 3

Beatriz de Jesus Costa

***Abstract*—Neste relatório realizamos a execução de dois códigos que resolver o mesmo problema, o contar 3. Um algoritmo foi feito usando a biblioteca pthread.h, o que o tornou paralelizável, enquanto o outro é sequencial. Foram calculados os speedups dos algoritmos para diversos números de threads e de entradas. O que se pôde concluir a partir da análise dos resultados foi que quanto maior o número de threads, maior o speedup.**

***Index Terms*—Programação paralela, CPU, GPU, paralelismo, multicore processors.**

I. Introdução

Problemas sequencias têm sido desenvolvidos desde o início de nosso aprendizado na computação, e está na hora de aprender a paralelizá-los. Tomamos o problema contar 3, que é feito de forma simples sequencialmente e construimos um código paralelo para ele, podendo assim dominuir seu tempo de execução utilizando diversas threads.

30 de Agosto, 2020

A. Problema contar 3

O problema contar 3 consiste em, dada uma quantidade de números de entrada, um vetor é preenchido aleatóriamente com essa quantidade de números e é preciso contar quantos números 3 se encontram nesse vetor. Também é possível inserir a porcentagem de números 3 que encontraremos neste vetor.

B. Algoritmo Paralelo

Na versão paralela do problema contar 3, podemos passar qualquer número de threads por parâmetro, e na execução do algoritmo cada uma delas será encarregada de um segmento do vetor, assim, cada thread lê somente uma parte do vetor e obtém o resultado de quantos números 3 havia naquela parte. As threads são criadas usando o comando pthread_create e depois são juntadas usando o comando pthread_join, para se obter o resultado final. Também foi preciso usar pthread_mutex_lock e pthread_mutex_unlock para evitar o problema de threads escreverem no 'count' ao mesmo tempo, alterando assim os resultados.

C. Especificações Técnicas

O problema foi executado nos modos paralelo e sequencial em uma máquina com 12GB de RAM e 4 núcleos de CPU, com frequência de 1,60GHz e 6MB de cache.

D. Experimentos e Resultados

Foram realizados experimentos com as entradas 100000000000, 10000000 e 10000. Variando também o número de threads e 1 a 8. O speedup indica quantas vezes o programa paralelo é mais rápido que a versão sequencial para executar uma dada tarefa, e foi calculado dividindo o tempo de processamento do algoritmo sequencial pelo tempo de processamento do algoritmo paralelo. A partir da figura 1 podemos perceber que com a entrada pequena, os resultados não são consistentes pois o tempo de execução foi muito pequeno (por volta dos 5 segundos), já quando aumentamos a entrada podemos verificar que quanto maior o número de threads, maior o speedup, ou seja, melhor é o algoritmo pararelo em relação ao sequencial em questão de tempo de execução.

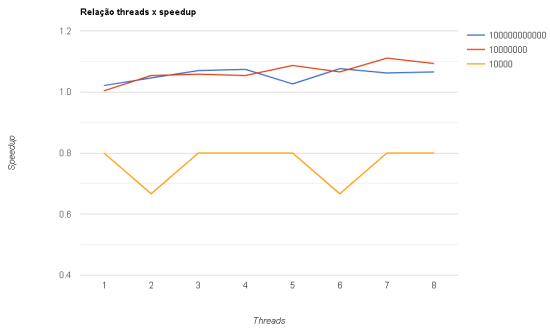

Fig. 1. Speedup por número de threads no algoritmo contar 3

II. Conclusion

A partir da criação e execução dos algoritmos e a realização dos diversos experimentos, podemos concluir que o algoritmo paralelo é melhor que o sequencial em questão de tempo de execução quando temos diversas threads. Também podemos concluir que a elaboração do algoritmo paralelo é mais complexa que a do sequencial, onde devemos levar em consideração diversos fatores para obtermos a execução e resultado corretos.