Universidade Federal de São Carlos

Programa de Pós-Graduação em Ciência da Computação

CCO-129-7 Introdução à Computação de Alto Desempenho (2021/2)

Exercício Programa - EP 3

OpenHPC UFSCar

Alcides Mignoso e Silva 760479

Submission Date : 07/03/2022

1 Atividade

O objetivo desta atividade é paralelizar a solução da equação de transferência de calor de Laplace pelo método de Jacobi utilizando **OpenMP**. O código base utilizado foi o existente em `https://github.com/HPCSys-Lab/HPC-101/tree/main/examples/laplace` e este relatório visa discutir valores de tempo de execução, speedup e eficiência das execuções.

As execuções foram realizadas utilizando matrizes com número de linhas e colunas igual a 2_000, e com 1, 2, 5, 10, 20 e 40 threads, em um computador com um processador AMD Ryzen 7 3700X @ 3.600GHz (16 threads) e 32gb de memória RAM.

2 Resultados

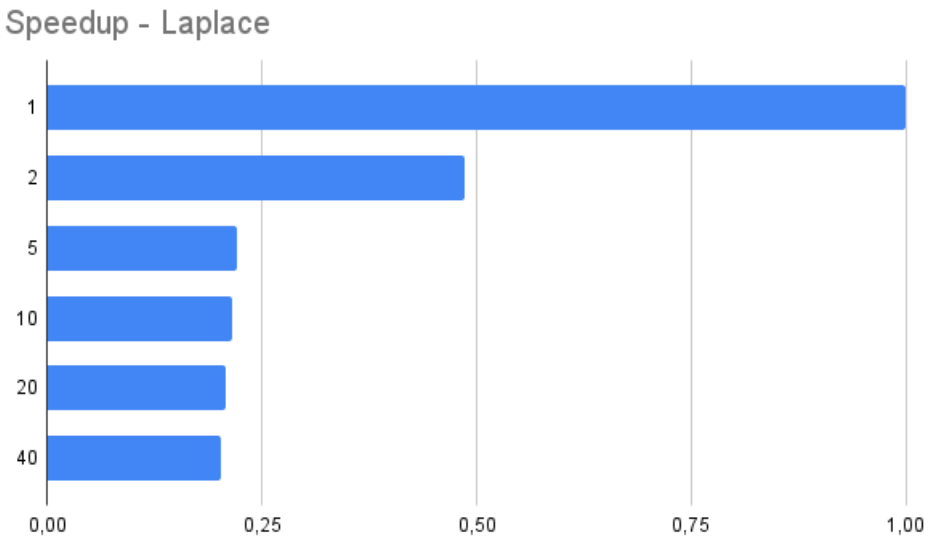
2.1 Tabelas

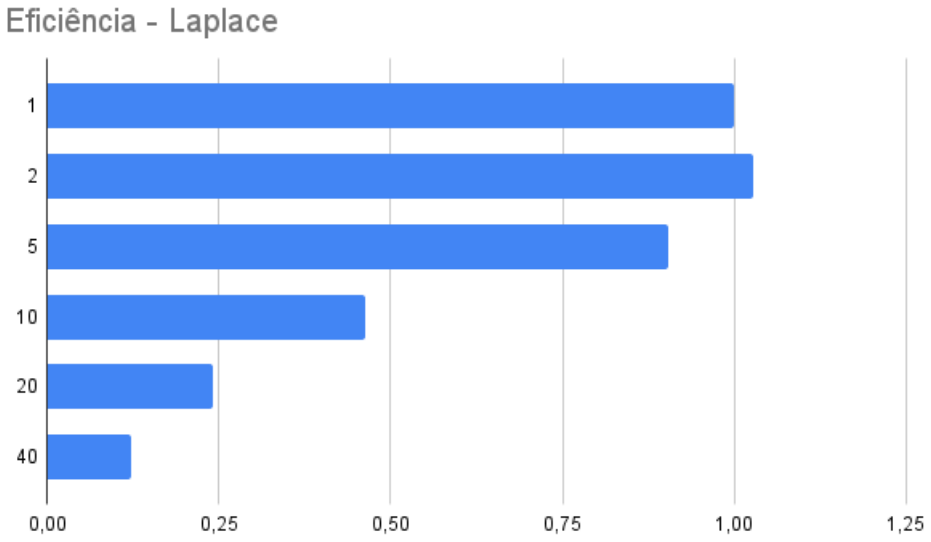
A tabela abaixo trás os tempos de execução do programa (em segundos) na versão sequencial e na paralelizada.

N. Threads	**Sequêncial**	**OpenMP**
1	100,6063	100,6063
2	100,6063	48,9180
5	100,6063	22,2578
10	100,6063	21,6874
20	100,6063	20,8471
40	100,6063	20,4186

2.2 Gráficos

Os gráficos abaixo ilustram valores de speedup e eficiência do programa (horizontal) em relação ao número de threads utilizadas na execução (vertical).





2.3 Discussão

Como pode-se perceber através do gráfico de speedup e da tabela de tempos de execução, a utilização de mais de 5 threads não impacta fortemente o tempo de execução do programa e consequentemente não altera bruscamente o speedup. Isso se dá por conta do overhead adicionado na criação de novas threads. Ainda assim, pôde-se notar um valor de eficiência interessante com a utilização de 2 e 5 threads.

Uma consequência desses overheads é a tendência de diminuição da eficiência quando novas threads são adicionadas, dado que os tempos de execução acabam não sendo reduzidos significantemente. Caso novas threads continuassem a serem adicionadas, é esperado que o tempo de execução comece a crescer por conta dos overheads já mencionados, ao invés de diminuir - paralelamente a isso a eficiência continuaria a cair.

É válido ressaltar que existe um loop aninhado no código, onde somente o loop mais externo é paralelizado. A paralelização de ambos os loops foram testados (através do "collapse" do OpenMP) mas por conta da grande quantidade de operações do loop mais externo, não notou-se vantagens na abordagem.