Universidade Presbiteriana Mackenzie  
Computação Paralela – Turma 05P11  
Alan Meniuk Gleizer – RA 10416804  
10/05/2025

Lab 7 – Ordenação em OpenMP e OpenACC

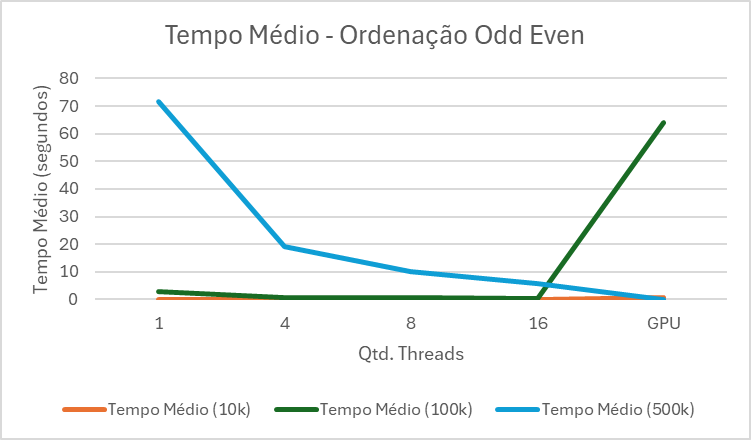
**Atividade 1 – Odd-Even Transposition Sort Paralelo**

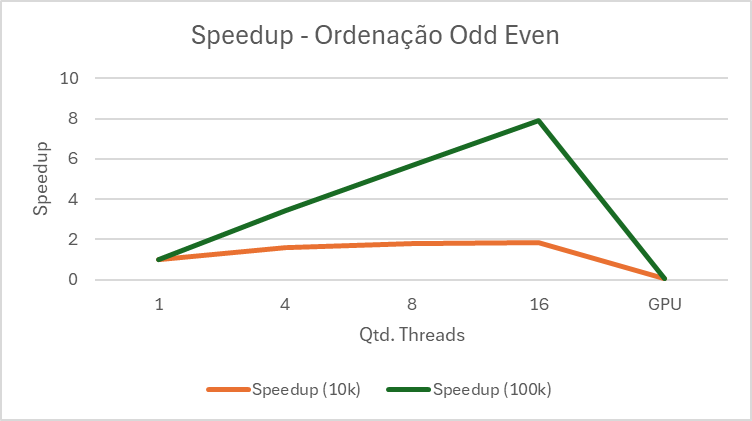
Prints execução

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

Análise dos Resultados

Para entender a efetividade das abordagens, foi necessário tabular os resultados obtidos, calcular o tempo médio para cada variação e o respectivo speedup. Os resultados podem ser observados abaixo:





O gráfico de speedup da Atividade 1 mostra um crescimento claro na performance à medida que o número de threads aumenta na versão OpenMP. Para vetores maiores (n = 500000), o speedup atinge valores expressivos com 16 threads, indicando que o algoritmo consegue aproveitar bem o paralelismo da CPU. No entanto, conforme aumentamos o número de threads, os ganhos marginais diminuem, o que é esperado devido ao aumento da sobrecarga de sincronização e ao fato de que Odd-Even Sort é sequencial em muitas de suas fases.

Já a versão com OpenACC apresenta desempenho drasticamente inferior. O gráfico de speedup colapsa: mesmo para vetores pequenos, o tempo de execução é dezenas de vezes maior que na versão OpenMP. Para n = 100000, o tempo excede 60 segundos. Isso acontece porque o algoritmo realiza trocas entre elementos vizinhos, o que gera dependências entre iterações e impede que o compilador paralelize os loops na GPU. Além disso, o custo de copiar dados entre host e device em cada fase contribui negativamente para o desempenho. Vale notar que o programa sequer apresentou resultados para n = 500000. Após 15 minutos, foi necessário interromper a execução.

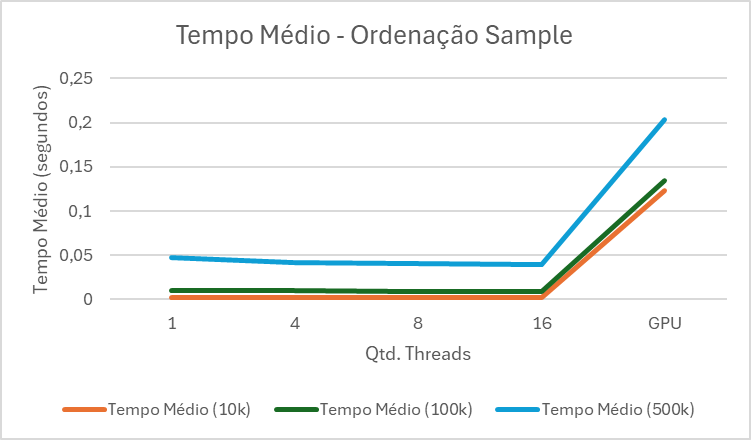
**Atividade 2 – Sample Sort Sort Paralelo**

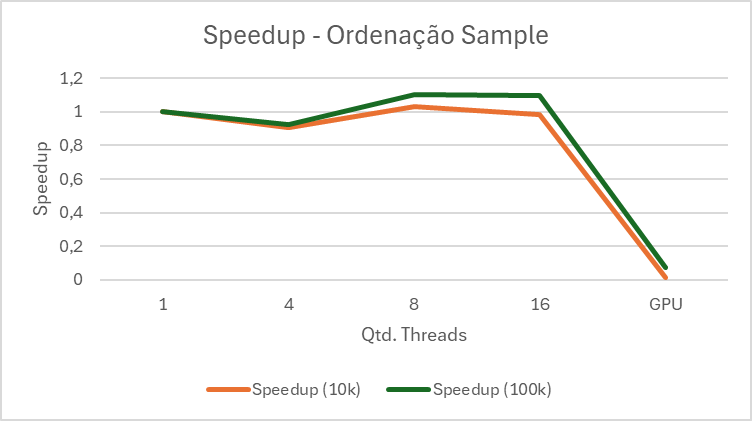
Prints execução

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

Análise dos Resultados

Para entender a efetividade das abordagens, foi necessário tabular os resultados obtidos, calcular o tempo médio para cada variação e o respectivo speedup. Os resultados podem ser observados abaixo:





O gráfico de speedup da Atividade 2 revela uma dinâmica diferente. A versão serial de Sample Sort já é rápida, especialmente para tamanhos de vetor menores, o que limita os ganhos potenciais com paralelismo. Nas versões com OpenMP, mesmo usando 16 threads, o speedup é modesto. Isso sugere que os trechos sequenciais do código dominam o tempo total, tornando o ganho com múltiplas threads pequeno em relação ao overhead do paralelismo.

A versão com OpenACC mostra tempos de execução significativamente maiores que a versão CPU, mesmo na versão paralela. O gráfico mostra que, para todos os tamanhos de entrada testados, o speedup da GPU em relação à versão serial da CPU é inferior a 1, ou seja, não há speedup. Isso provavelmente ocorre porque a GPU é utilizada apenas para ordenar os baldes no final, e o custo de transferir dados para a GPU acaba não se justificando. Com isso, mesmo sendo um algoritmo paralelizável em teoria, a forma como a divisão de tarefas foi feita limita os benefícios do uso da GPU. Vale notar que o algoritmo é complexo e certamente existem implementações mais eficientes, mas além das habilidades atuais.