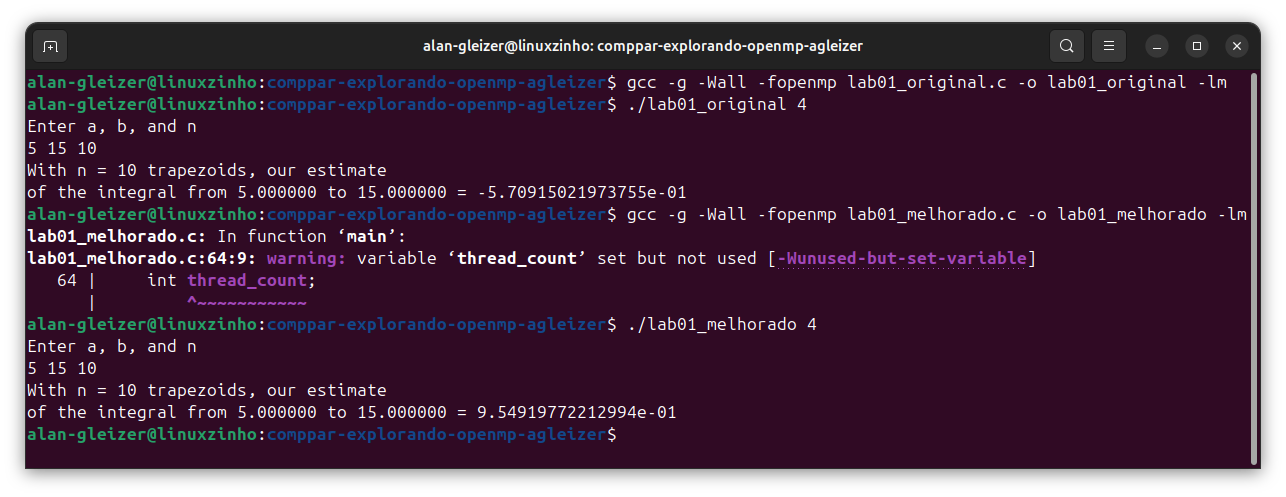
Universidade Presbiteriana Mackenzie  
Computação Paralela – Turma 05P11  
Alan Meniuk Gleizer – RA 10416804  
15/03/2025

Lab 4 – Introdução ao OpenMP

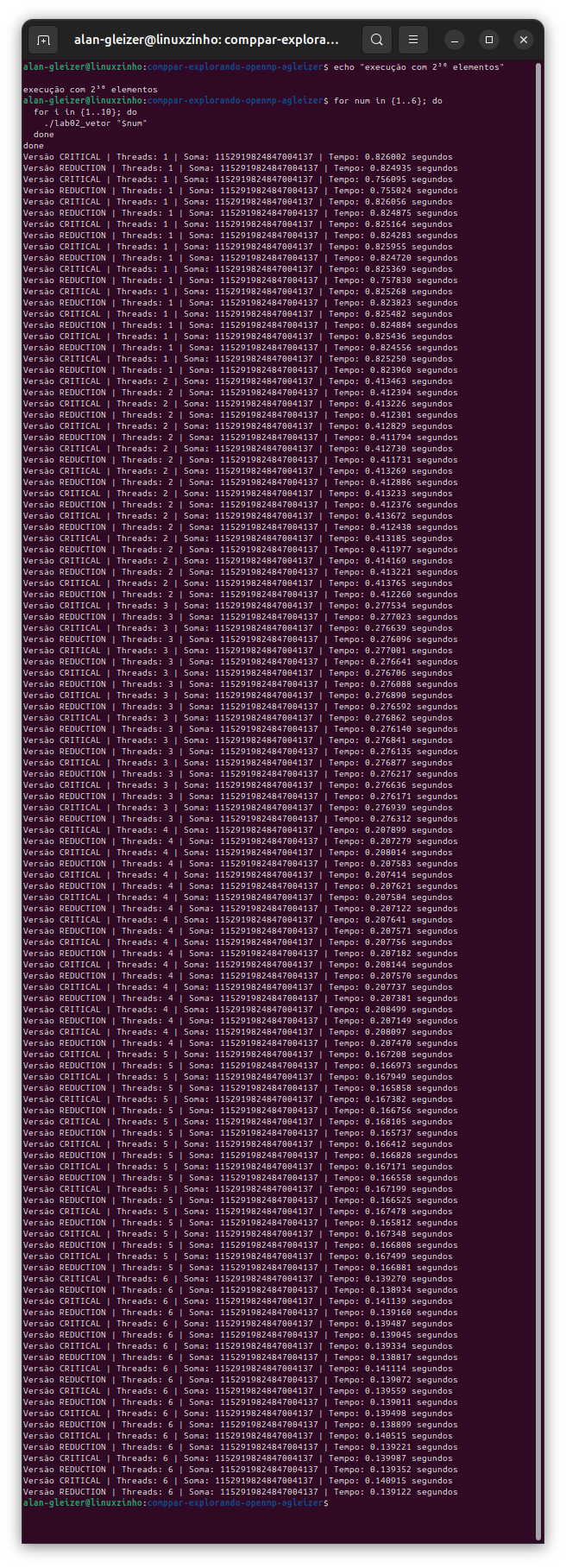
Lab 1 - Programação - Integral Regra do Trapézio

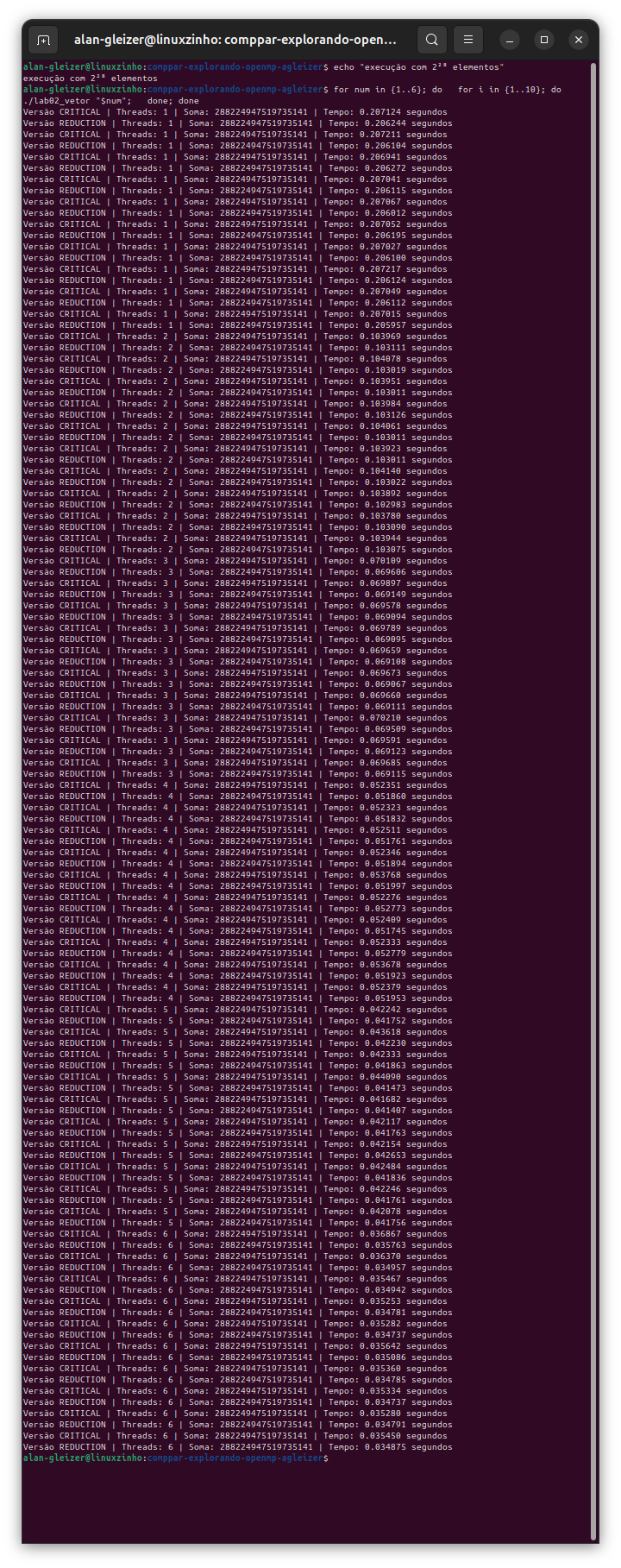
Print execução:

A number on a white background

AI-generated content may be incorrect.É possível observar que o resultado mudou. Verificando o resultado correto dessa integração em uma calculadora online temos que:

Assim, podemos afirmar que existia, na versão original, uma condição de corrida que causava um erro na avaliação da função, e esse erro foi corrigido com o uso do pragma reduction do OpenMP.

Lab 2 - Avaliação de desempenho  
Prints de execução:

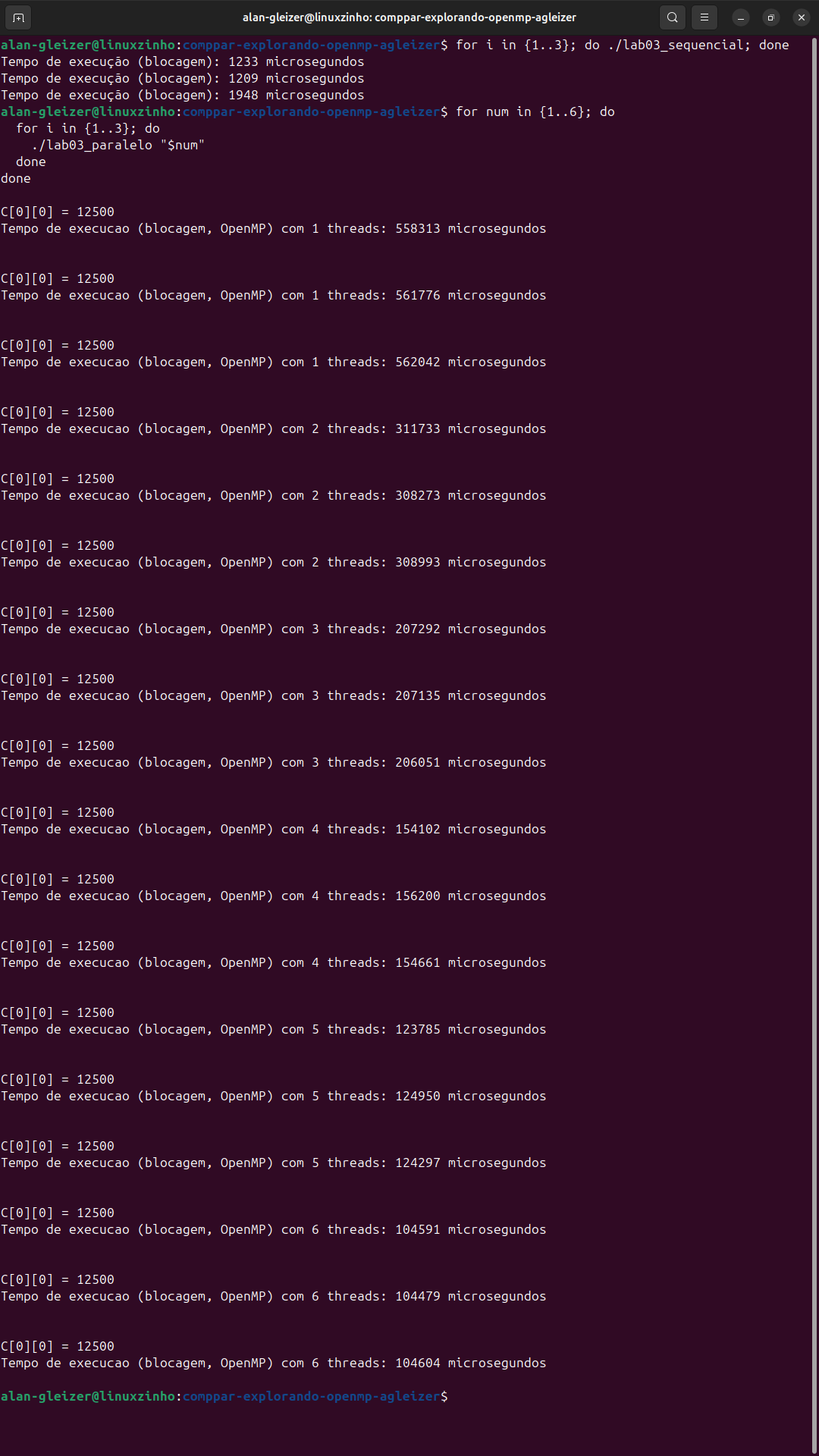


Gráficos comparativos:

**(obs. O excel com os dados abertos está neste mesmo diretório)**

Comparando as versões Critical e Reduction, os resultados mostram que ambas escalam bem até 6 threads, reduzindo significativamente o tempo de execução à medida que o número de threads aumenta. Ambas apresentam um Speedup QUASE linear. Vale observar que a versão Reduction foi consistentemente mais rápida que a Critical, provavelmente por evitar o overhead causado pelo uso de critical. Além disso, os testes com 228 230 elementos mostraram padrões semelhantes, indicando que a paralelização funciona bem independentemente do tamanho do problema, até os tamanhos analisados. No geral, a versão Reduction é mais eficiente e deve ser priorizada.

Lab 3 - - Nova multiplicação de matrizes

Print de execução:

Gráficos comparativos:

(obs. O excel com os dados abertos está neste mesmo diretório)

É possível observar o tempo muito alto da versão paralela com 1 thread acontece por causa do overhead da paralelização, ou seja, o custo extra para criar e gerenciar threads. Com apenas 1 thread, esse custo não compensa e a execução fica mais lenta do que a versão sequencial.

Porém, para matrizes muito grandes, esse overhead se torna pequeno em comparação com o tempo total de cálculo. Com mais threads, o trabalho é dividido e o ganho de desempenho compensa o custo inicial, como visto nos testes, onde 6 threads tornam a execução 5,36x mais rápida do que com 1 thread. O speedup é quase linear, sugerindo a possibilidade de escala do trabalho.