Projeto 1 Computação Paralela

Integrantes

Carlos Eduardo Rosendo Basseto – 10409941

João Rocha Murgel - 10410293

Luiz Henrique Ribeiro Pulga – 10409246

Introdução –

O objetivo deste projeto foi acelerar a análise de um arquivo de log de acesso web de grande volume. A abordagem sequencial, embora funcional, utiliza apenas um único núcleo do processador, deixando o restante da capacidade computacional do hardware moderna ociosa. Para superar essa limitação, foi implementada uma estratégia de paralelização utilizando multithreading com a biblioteca pthreads, aplicando o paradigma de Paralelismo de Dados.

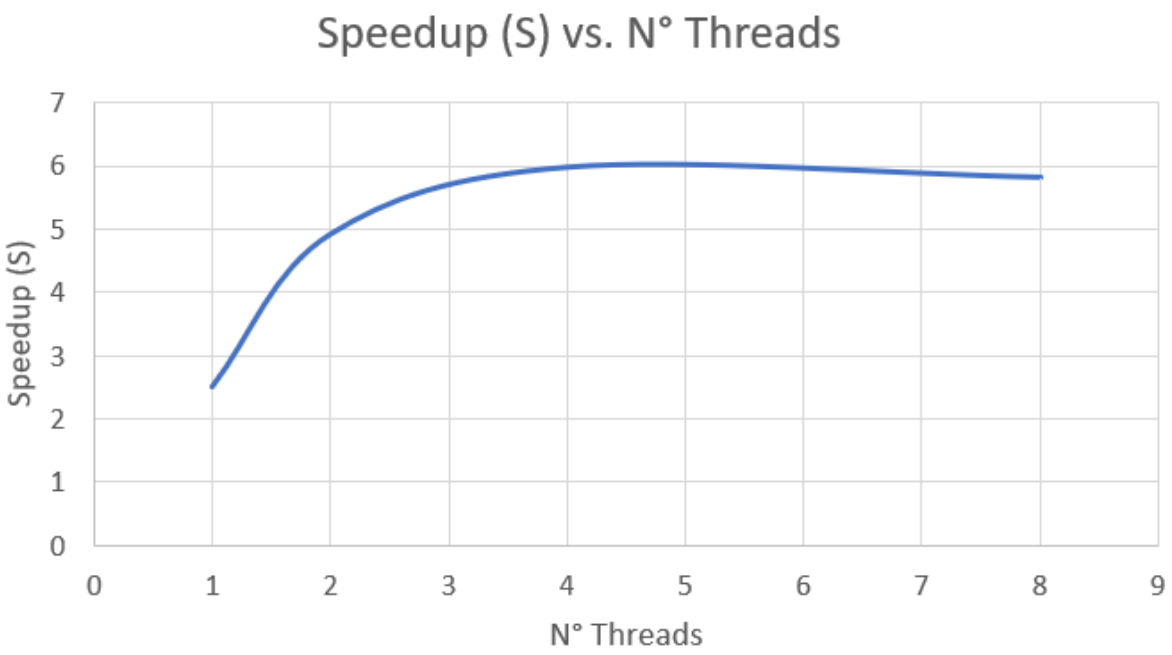
Inicialmente, o arquivo de log é inteiramente carregado em um array na memória. Este array é então dividido em blocos de linhas de tamanho similar, e cada bloco é atribuído a uma thread para processamento concorrente.

Para garantir a eficiência e evitar gargalos de sincronização, foi utilizada uma otimização crucial: cada thread calcula suas estatísticas erros 404 e bytes transferidos em variáveis locais e privadas. O acesso à estrutura de dados global, que armazena o resultado final, é protegido por um mutex. Esta trava é acionada apenas uma única vez por thread, ao final de todo o seu processamento, para agregar seus resultados parciais ao total. Essa abordagem minimiza a contenção e permite alta escalabilidade.

Tabela de Desempenhos –

N° Threads	Tempo de execução (s)	Speedup (S)	Eficiência (E)
1	0,1574	2,51778907	2,51778907
2	0,0805	4,92298137	2,46149068
4	0,0663	5,97737557	1,49434389
8	0,0681	5,81938326	0,72742291

Gráfico de Speedup vs. N° de Threads –



Conclusão –

Os resultados mostram que o speedup obtido não foi o esperado. Uma escalabilidade forte, ou speedup linear. No gráfico, a linha ideal seria uma reta ascendente de 45 graus

No entanto, o gráfico e a tabela revelam que o speedup aumenta rapidamente até cerca de 4 threads, atinge um pico, e depois começa a diminuir, de forma que o speedup para 8 threads é, na verdade, menor do que o speedup para 4 threads. Isso indica que, a partir de um certo ponto, o aumento no número de threads não resulta em um ganho de desempenho proporcional.

A principal razão pela qual o speedup não é linear é devido a dois fatores:

1. **A Lei de Amdahl:** Esta lei afirma que o ganho de desempenho é limitado pela porção do programa que não pode ser paralelizada, ou seja, a parte sequencial. Mesmo com muitas threads, o tempo gasto nesta parte do código se torna o principal gargalo, limitando o speedup total.
2. **Overhead de Sincronização:** O programa utiliza um mutex para proteger seções críticas. À medida que o aumento do número de threads, a competição por esse mutex se torna mais intensa. As threads perdem tempo esperando para acessar a seção protegida, o que consome o ganho de tempo da paralelização. O declínio no speedup para 8 threads é a prova de que este overhead de sincronização se tornou tão grande que prejudicou o desempenho geral, superando os benefícios de ter mais threads.