Trabalho de OPENMP: Gauss

Aluno: Pedro Antônio da Silva | E-mail: 194828@upf.br

1. Descrição da Implementação

Foi realizada uma análise de performance (*profiling*) no código sequencial com a ferramenta *gprof*, que demonstrou que a função ***solveLinearSystem*** consumia **em média** 99.3% do tempo computacional o que corresponde a 622,2 segundos, sendo o alvo principal para a otimização.

Dentro da função, o maior trabalho é realizado na fase de eliminação progressiva ($O(n^3)$), foco da paralelização. A fase de substituição retroativa, ($O(n^2)$) foi mantida sequencial devido à sua dependência de dados entre iterações.

Funções como as de leitura do sistema e escrita dos resultados não foram modificadas por não terem impacto significativo no ganho de eficiência.

2. Recursos de Programação Paralela Utilizados

A paralelização foi implementada de forma estratégica com a diretiva ***#pragma omp parallel for***, focado no laço de atualização das linhas da matriz, cujas iterações independentes permitem uma divisão segura e eficiente do trabalho. A correção do algoritmo foi garantida pela cláusula ***private***, essencial para evitar *race conditions* ao designar cópias exclusivas das variáveis de controle para cada *thread*. Para maximizar o desempenho, optou-se pelo ***schedule****(static),* que minimiza o *overhead* de escalonamento ao distribuir de forma fixa uma carga de trabalho já balanceada, assegurando a abordagem mais eficiente para a estrutura do problema.

3. Análise dos Resultados dos Testes

A análise dos resultados, detalhados na Tabela do **Anexo 1**, revela sucesso na implementação paralela, porém com uma escalabilidade não linear. Nota-se um speedup superlinear com 2 threads (eficiência média de 102%), o que pode ser atribuído a uma otimização no uso da hierarquia de memória cache que minimiza a latência de acesso à RAM. Já em 4 threads, embora não se mantenha o comportamento superlinear, a eficiência ainda é elevada (cerca de 94%), indicando bom aproveitamento dos recursos de paralelização. Porém, a partir de 8 threads, a performance entra em um regime de ganhos marginais decrescentes, onde a porção sequencial do código e o overhead de gerenciamento das threads, característica da Lei de Amdahl. O efeito piora em 16 threads, resultando em degradação de performance, na qual o tempo de execução aumenta devido à supersaturação de recursos de hardware e à contenção de memória.

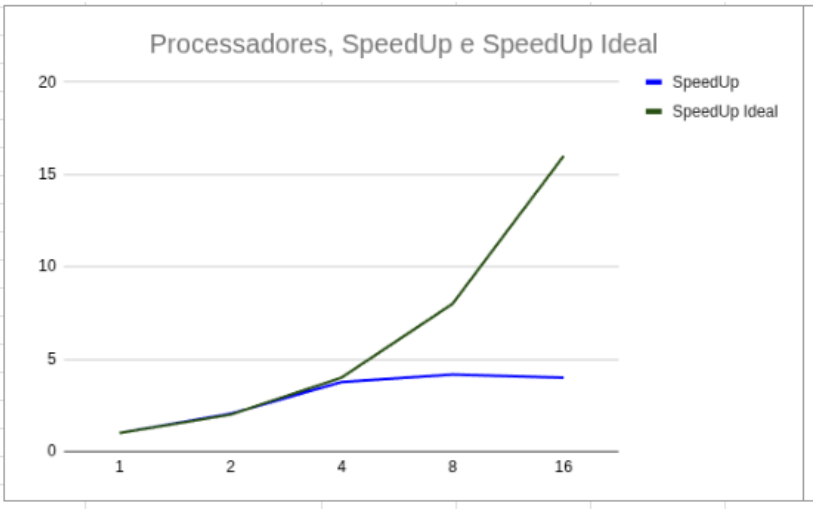
4. Considerações Finais

Conclui-se, portanto, que a configuração ótima para esta aplicação no sistema testado é de 4 threads, que representa o ponto de equilíbrio ideal entre o ganho de paralelismo e o custo do overhead.

Anexo 1: Tabela: Métricas de Desempenho:

Processadores	Tempo de execução paralelo	SpeedUp	Eficiência	Custo	SpeedUp Ideal
1	622,2	1	1	617	1
2	304,7	2,042008533	1,021004266	609,4	2
4	165,4	3,761789601	0,9404474002	661,6	4
8	149,4	4,164658635	0,5205823293	1195,2	8
16	155,8	3,993581515	0,2495988447	2492,8	16

Anexo 2: Gráfico: Processadores, SpeedUP e SpeedUP ideal:



Anexo 3: Gráfico: Processadores x Eficiência: