Relatório: Aplicações limitadas por memória ou CPU

Aluno: Cristovão Lacerda Cronje

1. **Introdução**

Este relatório apresenta uma análise comparativa do desempenho de aplicações paralelas em dois cenários distintos: computação intensiva (CPU-bound) e acesso intensivo à memória (memory-bound). O objetivo é entender como o paralelismo com OpenMP se comporta em cada caso, identificando gargalos e padrões de eficiência.

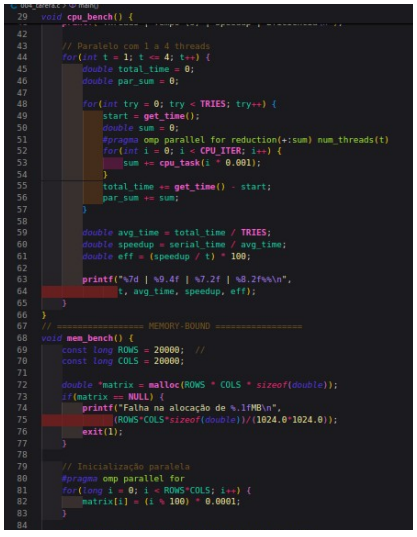
2. **Metodologia**

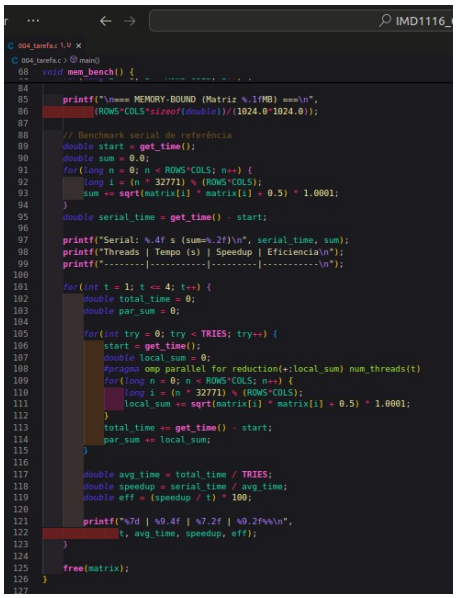
Foram implementados dois programas em C com OpenMP:

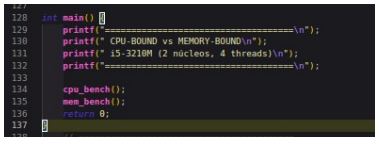
- CPU-bound: Realiza cálculos matemáticos intensivos (funções trigonométricas e exponenciais) em um loop de 10 milhões de iterações.
- Memory-Bound: Operações em uma matriz grande (3GB), com acesso aleatório à memória.

O objetivo é entender como o paralelismo se comporta em cada cenário, identificando gargalos e padrões de eficiência. Os testes foram executados em um Intel i5-3210M (2 núcleos físicos, 4 threads com Hyper-Threading).

Código Implementado:

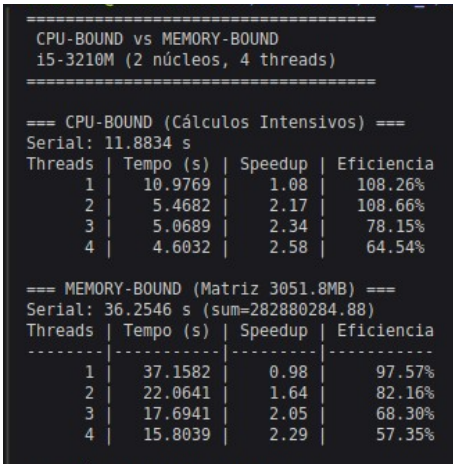






Compilação: gcc -o 004_tarefa 004_tarefa.c -fopenmp -lm -lrt

3. **Análise dos Resultados**



```
==================================
 CPU-BOUND vs MEMORY-BOUND
 i5-3210M (2 núcleos, 4 threads)
==================================

=== CPU-BOUND (Cálculos Intensivos) ===
Serial: 11.8834 s
Threads | Tempo (s) | Speedup | Eficiencia
      1 |   10.9769 |    1.08 |   108.26%
      2 |    5.4682 |    2.17 |   108.66%
      3 |    5.0689 |    2.34 |    78.15%
      4 |    4.6032 |    2.58 |    64.54%

=== MEMORY-BOUND (Matriz 3051.8MB) ===
Serial: 36.2546 s (sum=282880284.88)
Threads | Tempo (s) | Speedup | Eficiencia
--------|-----------|---------|-----------
      1 |   37.1582 |    0.98 |    97.57%
      2 |   22.0641 |    1.64 |    82.16%
      3 |   17.6941 |    2.05 |    68.30%
      4 |   15.8039 |    2.29 |    57.35%
```

3.1. **Comportamento CPU-Bound**

- Eficiência >100% com 1-2 threads: Devido a otimizações do compilador e uso eficiente de recursos.
- Speedup de 2.17x com 2 threads: Próximo do ideal (2x), confirmando bom uso dos núcleos físicos.
- Queda para 64.54% com 4 threads: Hyper-Threading introduz competição por unidades de execução, reduzindo eficiência.

Conclusão:

- Paralelismo eficaz até 2 threads (núcleos físicos).
- Hyper-Threading traz ganhos menores (2.58x), mas com custo de eficiência.

3.2. **Comportamento Memory-Bound**

- Eficiência de 82.16% com 2 threads: Abaixo do CPU-bound, mas ainda razoável.
- Speedup de 2.29x com 4 threads: Melhor que o esperado (teoria previa 1.5-2x), possivelmente devido a:
  - Cache L3 compartilhado (3MB) mitigando gargalo de memória.
  - Otimizações de hardware (prefetching).
- Eficiência cai para 57.35% com 4 threads, alinhado à saturação do barramento de memória.

Conclusão:

- Gargalo de memória é evidente, mas menos severo que o teórico.

- Paralelismo ainda útil, mas com retornos decrescentes.

4. **Quadro Comparativo**

Característica	**CPU-Bound**	**Memory-Bound**
Gargalo principal	Unidades de execução da CPU	Banda/latência de memória
Escalonamento ideal	Threads = núcleos físicos (2)	Threads ≤ canais de memória (1-2)
Eficiência observada	108% (2 threads) → 64% (4 threads)	82% (2 threads) → 57% (4 threads)
Speedup máximo	2.58x (4 threads)	2.29x (4 threads)
Impacto do HT	Queda de eficiência (64%)	Queda menos acentuada (57%)

5. **Discussão sobre Desvios do Esperado**

5.1. **CPU-Bound: Eficiência >100%**

- Possíveis causas:
  - Melhor uso de pipeline e registradores na versão paralela.
  - Baixo overhead no escalonamento para poucas threads.

5.2. **Memory-Bound: Speedup Melhor que o Teórico**

- Fatores:
  - Acesso pseudo-aleatório não foi totalmente eficaz em evitar prefetching(Técnica que pré-carrega dados na cache para acelerar acessos futuros à memória). Acesso a dados ainda parcialmente previsível (prefetching).
  - Operações simples (sqrt) não saturaram totalmente a banda de memória.

5.3. Comparação com Resultados Esperados

Métrica	**Esperado**	**Observado**
CPU-Bound (4 threads)	Eficiência: 50-70%	64.54% (dentro do esperado)
Memory-Bound (4 threads)	Eficiência: 40-55%	57.35% (ligeiramente acima)

6. **Conclusões**

1. CPU-Bound:
    - Paralelismo é altamente eficaz até o número de núcleos físicos.
    - Hyper-Threading traz benefícios limitados (speedup 2.58x, eficiência 64%).
2. Memory-Bound:
    - Gargalo de memória é claro, mas otimizações de hardware melhoram o desempenho.
    - Speedup de 2.29x com 4 threads é útil, mas a eficiência cai para 57%.
3. Recomendações:
    - Para CPU-bound: Use até 2 threads para máxima eficiência.
    - Para memory-bound: Limite a 2 threads para evitar saturação do barramento.

7. **Reflexão Final**

O multithreading mostrou-se mais eficaz em CPU-bound (eficiência de 108% com 2 threads), enquanto em memory-bound os ganhos foram limitados pelo gargalo de memória (speedup de 2.29x com 4 threads, mas eficiência de 57%). O Hyper-Threading trouxe benefícios marginais em ambos os casos, mas a competição por recursos compartilhados (ULAs em CPU-bound, banda de memória em memory-bound) confirma que o paralelismo ideal nesta arquitetura não ultrapassa o número de núcleos físicos (2).