Simulação de N-Corpos com MPI e OpenMP

Trabalho Prático 3 – Computação Paralela – 2025/2

Professor: Rafael Sachetto Oliveira

Davi Jannotti Coelho Pinheiro

Lucas de Lima Bergami

João Pedro Dani Laguardia

December 12, 2025

1 Introdução

Este relatório apresenta a implementação sequencial e paralela de uma simulação de N-Corpos, um problema clássico que calcula a evolução de um sistema de corpos sob a influência de forças gravitacionais mútuas. O objetivo é analisar o comportamento do programa, identificar gargalos por meio do *profiling* com `gprof`, e avaliar a estratégia de paralelização híbrida utilizando MPI (Message Passing Interface) para comunicação entre processos e OpenMP para paralelismo de memória compartilhada dentro de cada processo.

A simulação envolve o cálculo de forças $O(N^2)$ a cada passo de tempo, o que torna o problema computacionalmente intensivo e um excelente candidato para paralelização.

2 Perfil de Desempenho Sequencial

Para analisar o comportamento da versão sequencial, o código foi compilado com a flag `-pg` e executado com uma entrada de $N = 5000$ corpos e 20 passos de tempo.

```
gcc -pg -o nbody_seq_prof nbody_seq.c -lm
./nbody_seq_prof input_5000.txt output_5000.txt
gprof nbody_seq_prof gmon.out > prof_seq.txt
```

O trecho relevante do relatório gerado pelo `gprof` é mostrado abaixo:

```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call  name
100.01      5.69     5.69       20   284.52   284.52
   calculate_forces
  0.00      5.69     0.00       20     0.00     0.00
     update_particles
```

A tabela abaixo resume esses dados:

% Time	Cumulative	Self	Calls	Self (ms/call)	Total (ms/call)	Function
100.01	5.69	5.69	20	284.52	284.52	calculate_forces
0.00	5.69	0.00	20	0.00	0.00	update_particles

Table 1: Resumo do flat profile para a versão sequencial ($N = 5000$).

O `gprof` confirma que a função `calculate_forces` é responsável por praticamente 100% do tempo total de execução. Isso ocorre devido à complexidade quadrática do algoritmo, onde cada corpo interage com todos os outros. A função `update_particles`, por sua vez, possui complexidade linear $O(N)$ e seu tempo de execução é desprezível em comparação. Portanto, a estratégia de paralelização deve focar na distribuição do cálculo das forças.

3 Estratégia de Paralelização

A paralelização foi implementada utilizando uma abordagem híbrida MPI + OpenMP, visando explorar tanto o paralelismo distribuído (entre nós/processos) quanto o paralelismo de memória compartilhada (dentro de cada nó).

3.1 Decomposição de Dados (MPI)

Utilizou-se uma decomposição de domínio onde o conjunto de N corpos é dividido igualmente entre os P processos MPI. Cada processo é responsável por calcular as forças e atualizar as posições de seus N/P corpos locais.

3.2 Comunicação All-to-All

Para calcular a força resultante sobre um corpo, é necessário conhecer a posição e massa de **todos** os outros corpos do sistema. Como cada processo possui apenas uma parte dos corpos atualizados, é necessária uma comunicação global a cada passo de tempo.

Utilizou-se a função `MPI_Allgatherv` (ou `MPI_Allgather`) para que todos os processos recebam as posições e massas atualizadas de todos os corpos.

- Antes do cálculo de forças, cada processo envia seus dados locais e recebe os dados de todos os outros.
- Foi criada uma estrutura `ParticleData` contendo apenas massa, x e y, para minimizar o volume de dados trafegados, reduzindo o overhead de comunicação.

3.3 Paralelismo de Laço (OpenMP)

Dentro de cada processo MPI, o cálculo das forças para os corpos locais é paralelizado com OpenMP. O laço externo, que itera sobre os corpos locais, é decorado com `#pragma omp parallel for`. Isso permite que múltiplas threads (cores) trabalhem simultaneamente no cálculo das forças, aproveitando a arquitetura multicore dos processadores modernos.

4 Avaliação Experimental e Resultados Esperados

Devido a limitações no ambiente de execução (ausência de bibliotecas MPI instaladas), não foi possível coletar dados empíricos de desempenho para a versão paralela neste relatório. No entanto, apresenta-se abaixo uma análise teórica do desempenho esperado baseada na implementação realizada.

4.1 Tempos Sequenciais

Os tempos de execução sequencial para diferentes tamanhos de entrada (N) foram medidos:

Table 2: Tempos Sequenciais Medidos

N (Corpos)	Tempo (s)
1000	0.23
5000	5.70
10000	22.85 (Estimado)

*Nota: O tempo cresce quadraticamente com N, conforme esperado (5000 $\approx 5 \times 1000 \rightarrow 25 \times$ tempo).

4.2 Análise de Escalabilidade Esperada

A complexidade computacional é $O(N^2/P)$ por processo, enquanto a complexidade de comunicação é $O(N)$ (com `Allgather`).

- **Speedup**: Espera-se um speedup quase linear para valores grandes de N, onde o custo computacional $O(N^2)$ domina o custo de comunicação $O(N)$.
- **Eficiência**: Para N pequeno, a eficiência deve cair devido à latência da comunicação e overhead do MPI.
- **Híbrido vs Puro**: A abordagem híbrida tende a ser mais eficiente em clusters de nós multicore, pois reduz o número de processos MPI (diminuindo a comunicação inter-processos) e aproveita a memória compartilhada via OpenMP.

5 Conclusão

O perfilamento da aplicação sequencial confirmou que o cálculo de forças é o gargalo crítico da simulação de N-Corpos. A estratégia de paralelização híbrida MPI+OpenMP proposta ataca esse gargalo distribuindo a carga computacional e utilizando comunicação coletiva para satisfazer as dependências de dados. Embora a validação experimental completa não tenha sido possível no ambiente atual, a análise teórica indica que a solução é escalável, especialmente para sistemas com grande número de corpos, onde a razão entre computação e comunicação é favorável.