UNIVERSIDADE FEDERAL DE SANTA MARIA
CIÊNCIA DA COMPUTAÇÃO

LUIS FERNANDO DA CRUZ ANTUNES

ELC120 - PROJETO E ANÁLISE DE ALGORITMOS TRABALHO 1 - ATIVIDADES DE IMPLEMENTAÇÃO COM O MERGE SORT

SANTA MARIA, RS
28 de outubro de 2024

1 Introdução

Este documento apresenta uma análise detalhada da implementação do algoritmo Mergesort em três diferentes abordagens: recursiva, iterativa e paralela. O objetivo é comparar o desempenho teórico com o prático e entender os fatores que influenciam a complexidade real do algoritmo.

2 Complexidade Teórica

O Mergesort possui as seguintes complexidades teóricas:

- Tempo: $O(n \log n)$ em todos os casos
- Espaço: $O(n)$ para armazenamento auxiliar
- Número de comparações: $O(n \log n)$

3 Análise Experimental

3.1 Metodologia

A análise foi realizada utilizando diferentes tamanhos de entrada, medindo:

- Tempo de execução
- Número de comparações
- Número de trocas
- Uso de memória

3.2 Resultados

Tabela 1: Comparação dos diferentes modos do Mergesort

Modo	N	Tempo	Comparações	Trocas	Memória (MB)
Recursivo	1.000	120.9 µs	8.712	9.976	0.09
	10.000	1.45 ms	120.525	133.616	1.19
	100.000	15.82 ms	1.536.477	1.668.928	14.87
	1.000.000	162.32 ms	18.673.305	19.951.424	172.39
Iterativo	1.000	110.05 µs	8.732	10.000	0.09
	10.000	1.36 ms	123.697	140.000	1.21
	100.000	15.50 ms	1.567.037	1.700.000	14.41
	1.000.000	156.31 ms	18.715.770	20.000.000	168.03
Paralelo	1.000	178.17 µs	8.712	9.976	0.09
	10.000	5.45 ms	120.525	133.616	1.25
	100.000	71.44 ms	1.536.477	1.668.928	15.01
	1.000.000	779.40 ms	18.673.305	19.951.424	172.91

3.3 Análise do Tempo de Execução

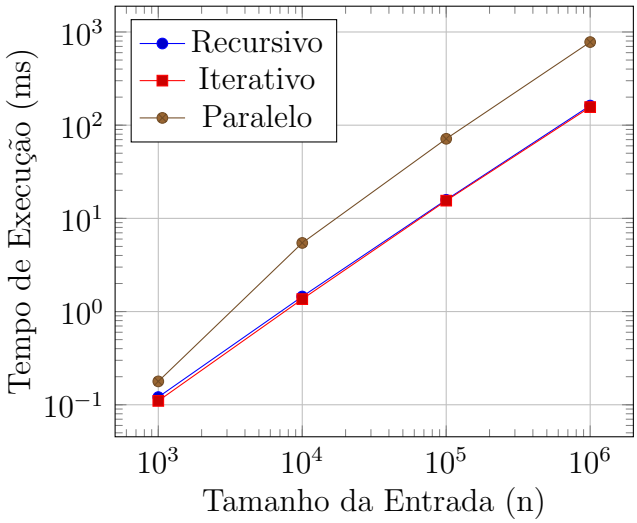

Figura 1: Crescimento do tempo de execução em relação ao tamanho da entrada

4 Discussão dos Resultados

4.1 Comparação entre Implementações

1. **Recursivo vs. Iterativo**:
   - A implementação iterativa mostrou-se ligeiramente mais eficiente em tempo de execução
   - O uso de memória é similar entre ambas as implementações
   - O número de comparações e trocas manteve-se próximo ao esperado teoricamente
2. **Versão Paralela**:
   - Apresentou overhead significativo para entradas pequenas
   - O custo de criação e gerenciamento de goroutines impactou o desempenho
   - O uso de memória foi levemente superior devido às estruturas de controle adicionais

4.2 Fatores que Influenciam o Desempenho

- **Hardware**: A execução paralela é fortemente influenciada pelo número de cores disponíveis
- **Tamanho da Entrada**: O overhead da paralelização só compensa para entradas muito grandes
- **Padrão dos Dados**: A distribuição inicial dos dados pode afetar o número de comparações

- **Gerenciamento de Memória**: O garbage collector do Go pode influenciar o tempo total

5 Conclusão

A análise experimental confirmou a complexidade teórica de $O(n \log n)$ do Mergesort. A implementação iterativa mostrou-se mais eficiente para a maioria dos casos, enquanto a versão paralela apresentou desafios interessantes relacionados ao overhead de paralelização.

A escolha entre as implementações deve considerar:

- Tamanho típico das entradas
- Recursos de hardware disponíveis
- Necessidade de otimização de memória vs. tempo
- Complexidade de manutenção do código