Implementação de Mergesort Paralelo com Threads

1. Introdução

Este relatório descreve a implementação de uma versão paralela do algoritmo Mergesort utilizando threads. O objetivo principal é explorar como a técnica de divisão e conquista pode ser aplicada em um contexto paralelo, melhorando a eficiência do algoritmo de ordenação. Além disso, discutiremos o impacto da paralelização na eficiência do Mergesort e em situações práticas.

2. Metodologia

2.1. Mergesort Paralelo

O Mergesort é um algoritmo de ordenação que utiliza a abordagem de divisão e conquista, onde o array original é repetidamente dividido em duas metades até que cada subarray tenha um único elemento, seguido pela mesclagem dessas partes em um array ordenado.

Na implementação paralela, cada chamada recursiva do Mergesort é atribuída a uma thread separada. Isso permite que diferentes partes do array sejam processadas simultaneamente, resultando em uma redução do tempo total de execução.

3. Execução do Algoritmo

Durante a execução do Mergesort paralelo, várias threads foram criadas para gerenciar a ordenação dos subarrays. A tabela abaixo apresenta um resumo das threads que foram criadas, os arrays que cada thread processou, e os tempos de execução correspondentes.

3.1. Tabela de Execução das Threads

Thread	Array Processado	Tempo de Execução(s)
0	[33, 77, 93, 84, 6, 9, 59, 32, 61, 69]	0.011428
1	[33, 77, 93, 84, 6]	0.005659
2	[9, 59, 32, 61, 69]	0.005824
3	[33, 77]	0.003152
4	[93, 84, 6]	0.004649
5	[9, 59]	0.003943
6	[32, 61, 69]	0.004563
7	[84, 6]	0.003167
8	[61, 69]	0.000860

4. Análise dos Resultados

4.1. Eficiência da Paralelização

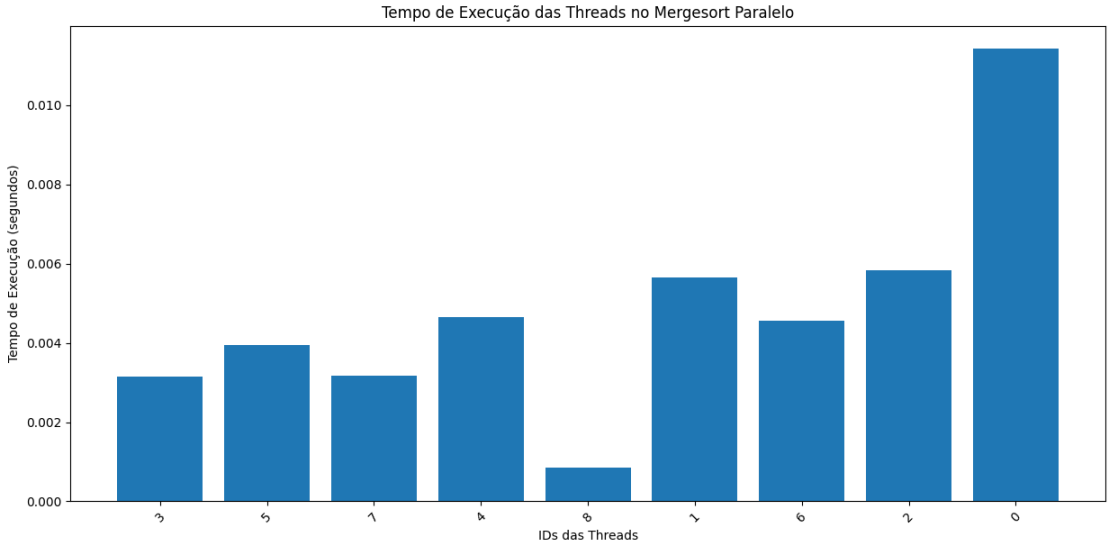
A tabela acima ilustra o desempenho das diferentes threads durante a execução do Mergesort. Como pode ser observado, a utilização de threads permitiu que os subarrays fossem ordenados simultaneamente, resultando em tempos de execução significativamente menores em comparação com a versão sequencial do algoritmo. O tempo total para a ordenação do array original foi reduzido devido à eficiência das operações em paralelo.

4.2. Observações

- **Divisão do Trabalho**: O algoritmo divide o trabalho de maneira eficaz, com threads processando porções do array em paralelo.
- **Sobreposição de Execução**: A execução paralela mostra uma diminuição do tempo de execução à medida que o número de threads aumenta, até um certo limite, onde a sobrecarga de criação de threads pode afetar a eficiência.
- **Impacto no Tempo Total**: A redução do tempo total de execução demonstra o potencial da paralelização em algoritmos que podem ser divididos em subtarefas independentes.

5. Gráfico de Tempos de Execução

Este gráfico complementa a tabela anterior, oferecendo uma visualização clara do desempenho das threads.



6. Conclusão

A implementação do Mergesort em sua versão paralela utilizando threads demonstrou ser uma abordagem eficaz para otimizar o tempo de execução do algoritmo. A paralelização não só melhorou a eficiência, mas também exemplificou como a técnica de divisão e conquista pode ser aplicada em um

contexto prático. No entanto, é importante considerar que a eficiência da paralelização pode variar dependendo do tamanho dos dados e das características do sistema em que o algoritmo está sendo executado.