Discente: Beatriz Saito Gobira

Docente: Alex Fernando de Araujo

Matriz Curricular: Processamento Paralelo.

É importante otimizar o desemprenho de algoritmos e processamento de dados, com isto devemos seguir boas práticas e evitar a problemas de ocorrência, bloqueio excessivos e ineficiências.

Problemas de ocorrência: acontece quando múltiplas threads ou processos estão tentando acessar, escrever ou modificar nos mesmos dados simultaneamente, isso pode gerar inconsistência e falta de sincronização nos dados.

Bloqueio excessivo: ocorre quando há sincronização excessiva entre as threads ou processos, podendo gerar baixo desempenho, concorrência reduzida, atraso de latência e entre outros.

Possíveis ajustes no escopo do código.

Encapsulamento e Passagem de Parâmetros: em vez de usar variáveis globais, é preferível encapsular as operações em funções e passar os parâmetros necessários explicitamente. Isso torna o código mais modular e fácil de entender.

*Uso de Semáforos ou Barreiras: e*m vez de usar um mecanismo simples de contagem de threads ativas, é recomendável utilizar semáforos ou barreiras para sincronizar a execução dos threads de forma mais robusta.

*Utilização de Pools de Threads: e*m vez de iniciar threads individualmente, é possível utilizar pools de threads, como concurrent.futures.ThreadPoolExecutor, para gerenciar automaticamente a alocação e execução das threads. Isso simplifica o código e melhora a eficiência.

*Otimização da Leitura e Escrita em Arquivos: p*ara evitar problemas de concorrência na escrita do arquivo de saída, é recomendável utilizar mecanismos de sincronização, como semáforos, ou considerar alternativas, como armazenar os resultados em memória e escrevê-los no arquivo de uma vez após a conclusão de todas as operações.

*Análise de Desempenho: p*ara garantir que as melhorias implementadas realmente resultem em ganhos de desempenho, é importante realizar uma análise detalhada do tempo de execução e do consumo de recursos do código antes e depois das alterações.

Possível ajuste geral do código.

Uma possível correção seria usar um único bloqueio para controlar o acesso ao arquivo de saída. Garantindo assim que apenas um thread pudesse escrever no arquivo, com isto resolveremos o problema de ocorrência, evitando condições repetidas.

Segue em anexo a correção do código conforme as possíveis soluções.