Desenvolvimento e avaliação de uma aplicação concorrente em `C`

O objetivo desta atividade é transformar uma aplicação sequencial (multiplicação de matrizes) em uma aplicação concorrente usando a linguagem `C` e a biblioteca `Pthread`, e avaliar o ganho de desempenho obtido. Para cada atividade, siga o roteiro proposto e responda às questões colocadas.

Atividade 1

Objetivo: Implementar uma solução sequencial para multiplicação de matrizes em `C` e coletar informações sobre o seu tempo de execução.

Roteiro:

1. Caso você não tenha implementado uma versão sequencial para o problema de multiplicação de matrizes, use o programa `multmat.c` como referência. Leia o código do programa e tente entender como ele funciona (atenção para as funções de coleta de tempo de processamento).
2. Se você já tem uma versão sequencial do problema implementada, excelente! Veja no arquivo `multmat.c` como incluir tomadas de tempo para avaliar o desempenho do seu programa.
3. Acompanhe a explanação do professor.
4. Compile e execute o programa sequencial, no mínimo três vezes, para cada uma das matrizes de entrada disponibilizadas no diretório dados. Qual parte do programa consome mais tempo?
5. Observe como os tempos medidos variam de acordo com a dimensão das matrizes de entrada.
6. Use a Lei de Amdahl para estimar qual será o ganho de desempenho máximo se a parte de multiplicação das matrizes for paralelizada, considerando uma máquina com dois processadores e as matrizes 2048×2048.

Relatório da atividade: Em um arquivo `.txt` registre as medidas de tempo coletadas a partir da execução do programa `multmat.c` e o cálculo da estimativa de ganho de desempenho de uma versão concorrente.

Atividade 2

Objetivo: Transformar a aplicação sequencial em uma aplicação concorrente.

Roteiro:

1. Altere o código `multmat.c` para que a parte de multiplicação da matriz seja feita por uma *thread* separada (a carga das matrizes de entrada e a impressão da matriz de saída devem continuar sendo implementadas pela função `main`).
2. Compile o programa `multmat.c` acrescentando a opção `-lpthread` na linha de comando (a opção `lpthread` acrescenta as funções da biblioteca *pthread*, ex.: `gcc -o multmat multmat.c -lpthread -Wall`).
3. Execute o programa no mínimo três vezes coletando os tempos medidos. Ocorreu alguma alteração em relação ao tempos coletados na versão original da aplicação? Por que?

Relatório da atividade: No mesmo arquivo `.txt` da atividade anterior, acrescente as medidas de tempo coletadas para essa versão do programa.

Atividade 3

Objetivo: Aumentar o número de *threads* da aplicação.

Roteiro:

1. Altere o código da atividade anterior fixando o número de *threads* em 2 e dividindo a tarefa de multiplicação igualmente entre elas (a carga das matrizes de entrada e a impressão da matriz de saída devem continuar sendo implementadas pela função `main`).
2. Execute o programa no mínimo três vezes coletando os tempos medidos. Ocorreu alguma alteração em relação ao tempos coletados na versão anterior da aplicação? Quais e por que?
3. Use a Lei de Amdahl para calcular qual foi o ganho de desempenho obtido para o caso das matrizes 2048×2048.

Relatório da atividade: No mesmo arquivo .txt da atividade anterior, acrescente as medidas de tempo coletadas para essa versão do programa e o cálculo do ganho de desempenho obtido com a versão concorrente.