# Laboratório 2
# Implementação e avaliação de aplicações concorrentes (parte 1)

**Computação Concorrente (MAB-117)**
**Prof. Silvana Rossetto**

[1]DCC/IM/UFRJ — 8 setembro de 2016

## Introdução

O objetivo deste Laboratório é aprender a medir o tempo de execução de trechos distintos de uma programa, implementar uma versão concorrente para o problema de multiplicação matriz-vetor e calcular e ganho de desempenho desse programa. Usaremos a linguagem C e a biblioteca *Pthreads*.

**Para cada atividade, siga o roteiro proposto e responda às questões colocadas.**

## Atividade 1

**Objetivo:** Aprender a realizar tomadas de tempo de execução de um programa em C.

**Roteiro:**

1. Abra o arquivo `incrementa_vetor.c` e veja como usar a função `GET_TIME()` (definida no arquivo `timer.h`) para coletar o tempo de processamento de cada trecho do programa. (Acompanhe a explanação da professora.)
2. Execute esse programa várias vezes, com 1, 2 e 4 threads, variando o número de elementos do vetor de $10^5$ a $10^9$, e observe os tempos medidos.
3. **A partir de qual tamanho do vetor de entrada o tempo de execução com mais de uma thread é menor que o tempo de execução com uma thread?**
4. Abra o arquivo `incrementa_vetor_bloco.c` e verifique como a estratégia de divisão da tarefa principal foi alterada. Que diferenças no padrão de acesso à memória essas duas estratégias apresentam?
5. **Repita a mesma sequência de passos para execução dessa nova versão da aplicação e compare os resultados obtidos.**

## Atividade 2

**Objetivo:** Compreender e experimentar uma solução sequencial para o problema de multiplicação matriz-vetor em C e coletar informações sobre o seu tempo de execução.

**Roteiro:**

1. Abra o arquivo `mult_matriz_vetor.c` e compreenda sua implementação.
2. As dimensões e os valores da matriz de entrada e do vetor de entrada deverão ser lidos de arquivos texto (exemplos de arquivos de entrada estão disponíveis na pasta `dados`). A primeira linha do arquivo deverá conter as dimensões da matriz (ou do vetor) e as linhas seguintes os valores dos seus elementos.
3. Acrescente ao programa chamadas da função `GET_TIME()` para **medir separadamente** os tempos de execução para **inicializar as estruturas de dados**, **executar a multiplicação** e **finalizar o programa**.

4. Execute o programa para cada uma das matrizes e vetores de entrada disponíveis no diretório `dados`.
5. **Observe como os tempos medidos variam de acordo com a dimensão das matrizes de entrada.**

## Atividade 3

**Objetivo:** Transformar a aplicação sequencial em uma aplicação concorrente.

**Roteiro:**

1. Altere o código sequencial para que a parte de multiplicação da matriz-vetor seja feita por uma ou mais threads separadas (a carga dos dados de entrada e a impressão do vetor de saída devem continuar sequenciais).
2. Execute o programa no mínimo **três vezes** coletando os tempos medidos.
3. Use a Lei de Amdahl para calcular qual foi o ganho de desempenho obtido.