Universidade Federal Rural do Rio de Janeiro

DCC - SmartCOMM

TM435 - PROGRAMAÇÃO PARALELA E DISTRIBUÍDA

Instruções

- A proposta dos exercícios é desenvolver o raciocínio lógico para aplicações que executam em arquiteturas paralelas/distribuídas.
- Para cada exercício é disponibilizado o seu código sequencial. Sendo assim, cada exercício contém o projeto fonte sequencial e uma instância de execução (exemplo), que pode ser utilizada para entender o funcionamento da computação do problema. A versão paralela deverá ser escrita a partir da versão sequencial.
- A versão paralela deverá receber as mesmas entradas e retornar a mesmas saídas do código sequencial. O desempenho do seu código paralelo, s (*speed-up*), dever ser calculado como segue: $s = \frac{T_s}{T_p}$, onde T_s é o tempo sequencial, T_p é o tempo sequencial.
- Para avaliar seu código paralelo sugerimos que criem mais instâncias de testes.
- Dúvidas poderão ser colocadas no Moodle e serão respondidas por um dos professores ou monitores.

Exercícios

Exercício 1: Autômato Celular é uma técnica capaz de reproduzir diversos problemas dinâmicos e complexos. Um desses problemas é conhecido como *game of life*, composto por células que formam uma grade regular de tamanho $m \times n$. Cada célula só pode assumir um dos dois estados (vivo ou morto) e deve obedecer o seguinte conjunto de regras:

1. Uma célula viva morre de solidão se tem menos de dois vizinhos vivos.
2. Uma célula viva morre de superpopulação se tem mais de três vizinhos vivos.
3. Uma célula morta torna-se viva se tem exatamente três vizinhos vivos.
4. Uma célula viva permanece viva se tem dois ou três vizinhos vivos.

A vizinhança adotada é ilustrada pela Fig. 1. A célula central é a analisada com oito células vizinhas e considere 0 na borda como condição de contorno.

Construa uma versão paralela conforme o módulo, usando como base o projeto: `game_of_life.tar.gz`.

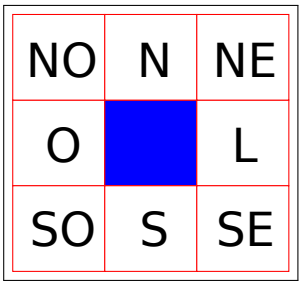

Figura 1: Vizinhança de Moore.

Exercício 2: A solução de uma integral é importante para alguns problemas. Dentre os métodos numéricos para a resolver a integração, o método dos trapézios é bem eficiente, onde o resultado aproximado da integral é dado pela soma das áreas dos n trapézios, cada qual definido pelo seu sub-intervalo, conforme ilustrado pela Fig. 2. O Erro numérico está diretamente relacionado ao Δx, pois quanto menor for o Δx, menor é o erro numérico. A Eq. 1 mostra a solução da integração.

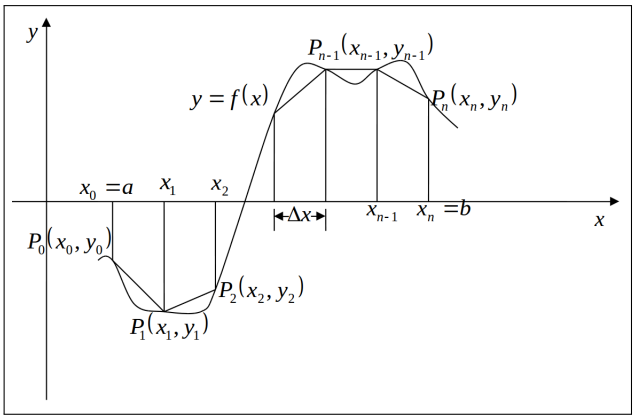

Figura 2: n trapézios no intervalo $[a, b]$.

$$I = \int_{x_0}^{x_n} f(x)dx \approx \frac{h}{2}\left[f(x_0) + 2\left[f(x_1) + f(x_1) + f(x_2) + \cdots + f(x_{n-1})\right] + f(x_n)\right] \tag{1}$$

Desenvolva a versão paralela conforme o módulo, usando como base o projeto: `integral.tar.gz`.

Exercício 3: Um das ferramentas utilizadas na computação científica é um resolvedor de sistema linear. Existem alguns métodos que tratam esse problema, dentre esses, o método de Jacobi recebe destaque por ser naturalmente paralelizável. Seu funcionamento é bem simples: re-escreve o sistema $Ax = b$ em $x = Fx + d$, onde:

- F é uma matriz $n \times n$ e,
- d é um vetor ($d \in R^n$).

- Parte-se de um valor inicial para x, onde $x \in R^n$

Implemente a versão paralela conforme o módulo, usando como base o projeto: `jacobi.tar.gz`.

Exercício 4: Uma das aplicações mais utilizadas na computação de alto desempenho é a simulação conhecida como **n-corpos**. É um problema de complexidade $O(n^2)$, onde n é a quantidade de corpos. Em linhas gerais, é calculada posição de cada corpo, considerando a influência de todos os outros corpos. É um tipo de aplicação que também é utilizada para simular o comportamento de partículas, galáxias e até mesmo multidão.
Construa uma versão paralela conforme o módulo, usando como base o projeto: `n-corpos.tar.gz`.