

Programação Paralela

Prof. Rodrigo Martins

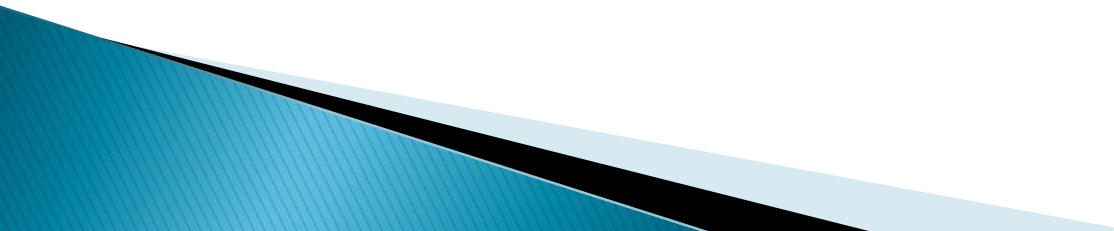
rodrigo.martins@francomontoro.com.br



Cronograma da Aula

- ▶ Apresentação da disciplina
- ▶ Critérios de Avaliação
- ▶ Metodologias de Trabalho
- ▶ Introdução a Programação Paralela

Ementa

- ▶ Modelos de computação paralela.
 - ▶ Expressão e extração do paralelismo.
 - ▶ Sincronização e comunicação: métodos e primitivas.
 - ▶ Programação concorrente e distribuída: linguagens e algoritmos.
 - ▶ Problemas clássicos de programação paralela. Princípios de implementação.
- 

Objetivos

- ▶ Capacitar o aluno a compreender os princípios de programação de aplicações paralelas e de processamento paralelo e desenvolver soluções usando esses princípios.

Critérios de Avaliação

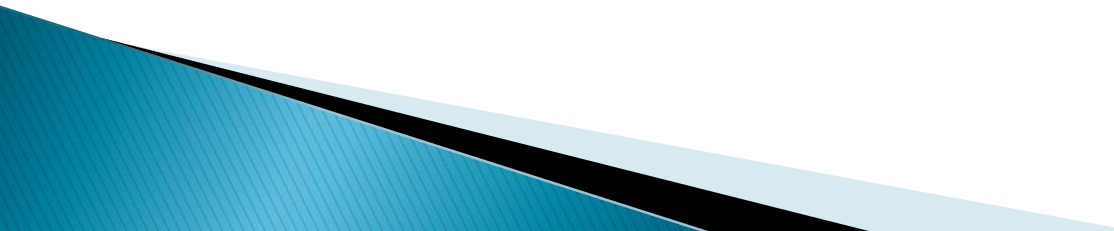
- ▶ T1 – Lista de Exercícios 1 (30%)
- ▶ P1 – Avaliação Bimestral 1 (70%)

- ▶ T2 – Lista de Exercícios 2 (30%)
- ▶ P2 – Avaliação Bimestral 2 (70%)

- ▶ Média Final – $(MB1 + MB2) / 2$

- ▶ Média final maior ou igual a 7,0 (sete) implicará em aprovação sem exame final;
- ▶ Média final igual ou superior a 4,0 (quatro) e inferior a 7,0 (sete) dependerá de aprovação em exame final;
- ▶ Média final de aproveitamento inferior a 4,0 (quatro) implicará em reprovação;
- ▶ A aprovação em exame final será obtida se a média aritmética da média final de aproveitamento com a nota do exame final for igual ou superior a 5,0 (cinco).

Metodologias de Trabalho

- ▶ Material exposto em sala de aula (Apresentações);
 - ▶ Indicação de Sites sobre o conteúdo (Artigos);
 - ▶ Pesquisas;
 - ▶ Uso de metodologias ativas no desenvolvimento de projetos.
- 

Bibliografia Básica

- DE ROSE, César A. F.; NAVAUX, Philippe O. A. Arquiteturas Paralelas. Porto Alegre: Sagra-Luzzato, 2008.
- ROOSTA, Seyed H. Parallel Processing and Parallel Algorithms: Theory and Computation. New York: Springer-Verlag, 2000.

Bibliografia Complementar

- GEI, Al et al. PVM: Parallel Virtual Machine, a user guide and tutorial for networked parallel computing. Cambridge, Massachusetts: MIT Press, 1994.
- PITANGA, Marcos. Construindo Supercomputadores com Linux. Brasport, 2002.
- WILKINSON, Barry and Allen, Michael. Parallel Programming. Techniques and Applications Using Networked Workstations and Parallel Computers. Prentice Hall. 1999. 1a. Edição.


Porque Programação Paralela?

- Porque permite que os computadores realizem tarefas mais rapidamente e com mais eficiência.
- Utiliza múltiplos núcleos de processamento para executar tarefas simultaneamente, em vez de executá-las de forma sequencial em um único núcleo.

Porque Programação Paralela?

- Dois dos principais motivos para utilizar programação paralela são:
 - Reduzir o tempo necessário para solucionar um problema.
 - Resolver problemas mais complexos e de maior dimensão.

Porque Programação Paralela?

- Outros motivos são:
 - Tirar partido de recursos computacionais não disponíveis localmente ou subaproveitados.
 - Ultrapassar limitações de memória quando a memória disponível num único computador é insuficiente para a resolução do problema.
 - Ultrapassar os limites físicos de velocidade e de miniaturização que atualmente começam a restringir a possibilidade de construção de computadores sequenciais cada vez mais rápidos.
- 

Porque Programação Paralela?

- Tradicionalmente, a programação paralela foi motivada pela resolução/simulação de problemas fundamentais da ciência/engenharia de grande relevância científica e econômica, denominados como **Grand Challenge Problems (GCPs)**.

Porque Programação Paralela?

- Tipicamente, os GCPs simulam fenômenos que não podem ser medidos por experimentação:
 - Fenômenos climáticos (e.g. movimento das placas tectônicas)
 - Fenômenos físicos (e.g. órbita dos planetas)
 - Fenômenos químicos (e.g. reações nucleares)
 - Fenômenos biológicos (e.g. genoma humano)
 - Fenômenos geológicos (e.g. atividade sísmica)
 - Componentes mecânicos (e.g. aerodinâmica/resistência de materiais em naves espaciais)
 - Circuitos eletrônicos (e.g. verificação de placas de computador)

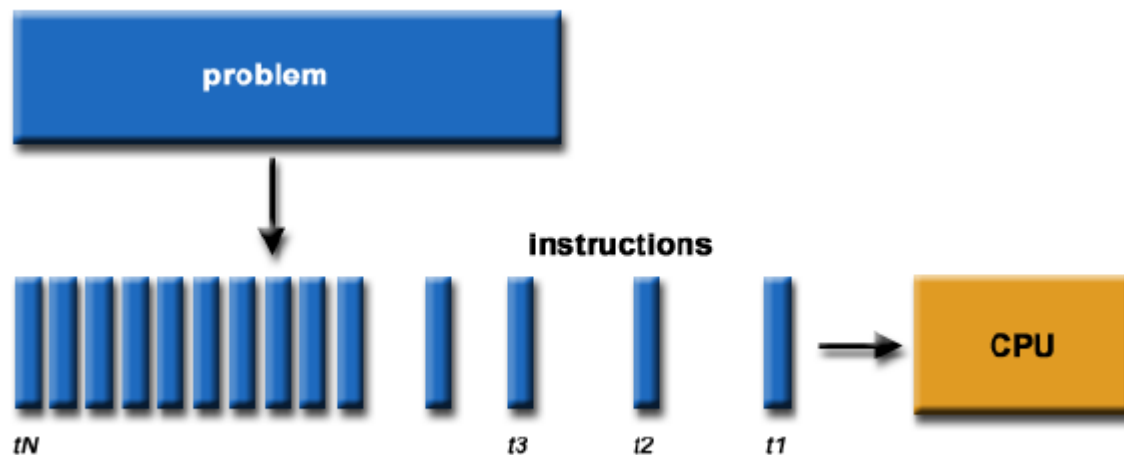
◦ ...

O que é Programação Paralela?

- É um paradigma de programação que visa a execução simultânea de várias tarefas computacionais em diferentes núcleos de processamento, como CPUs ou GPUs.

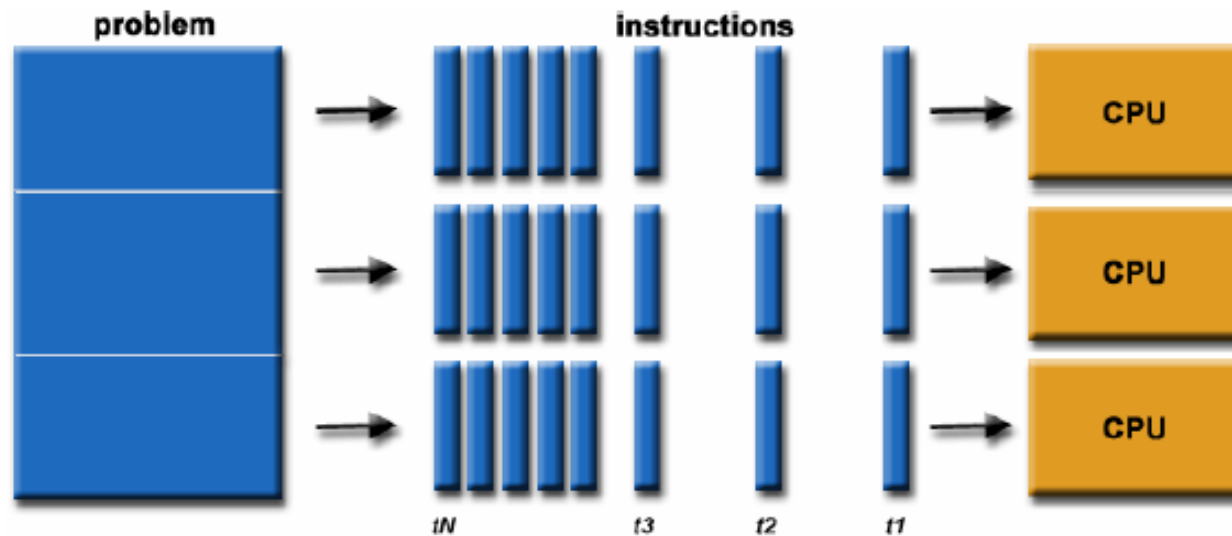
Programação Paralela X Programação Sequencial

- Na **programação sequencial**, as tarefas são executadas de forma sequencial, uma após a outra, em um único núcleo de processamento.
- Cada instrução é executada em ordem, uma após a outra.

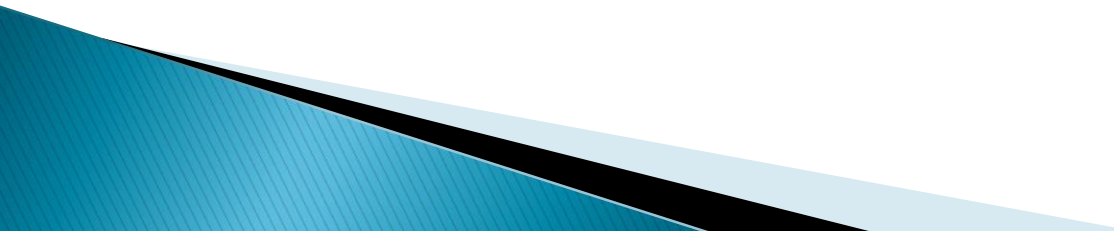


Programação Paralela X Programação Sequencial

- Na **programação paralela**, as tarefas são executadas simultaneamente em múltiplos núcleos de processamento.
- Isso permite que várias partes de um programa sejam executadas ao mesmo tempo, o que pode levar a uma execução mais rápida e eficiente.

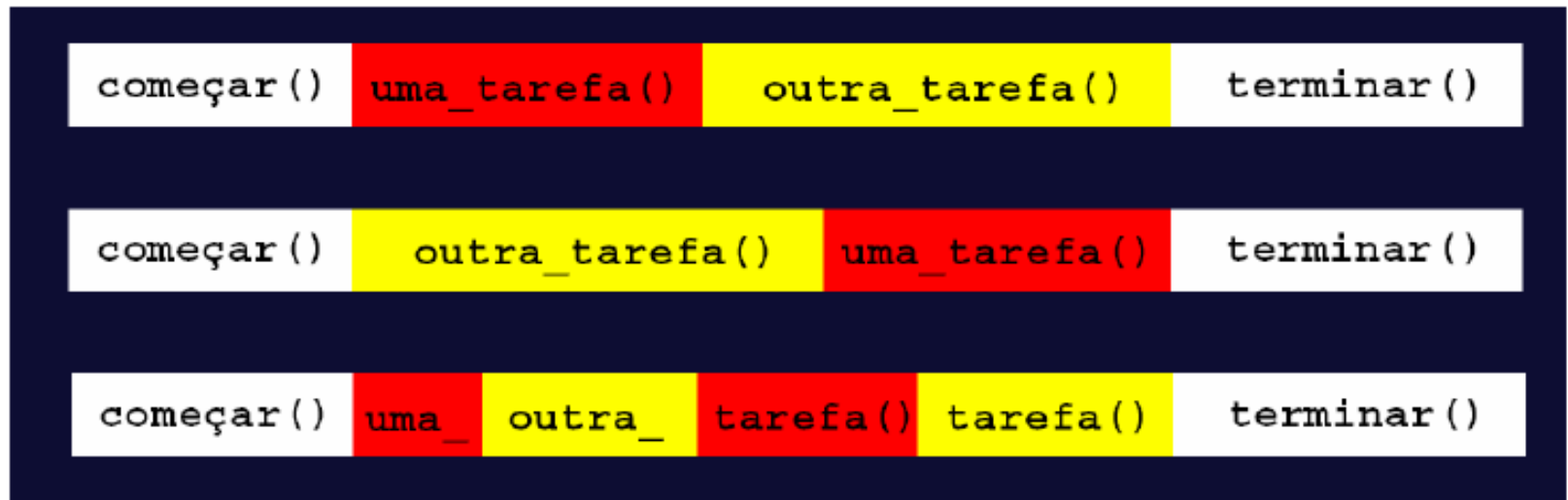


Programação Paralela X Programação Sequencial

- A programação paralela geralmente exige mais esforço e habilidade do programador do que a programação sequencial, porque a execução simultânea de várias tarefas pode levar a problemas de sincronização e comunicação entre os diferentes threads ou processos em execução.
 - Porém, quando feita corretamente, a programação paralela pode fornecer um grande aumento no desempenho e na eficiência em relação à programação sequencial.
- 

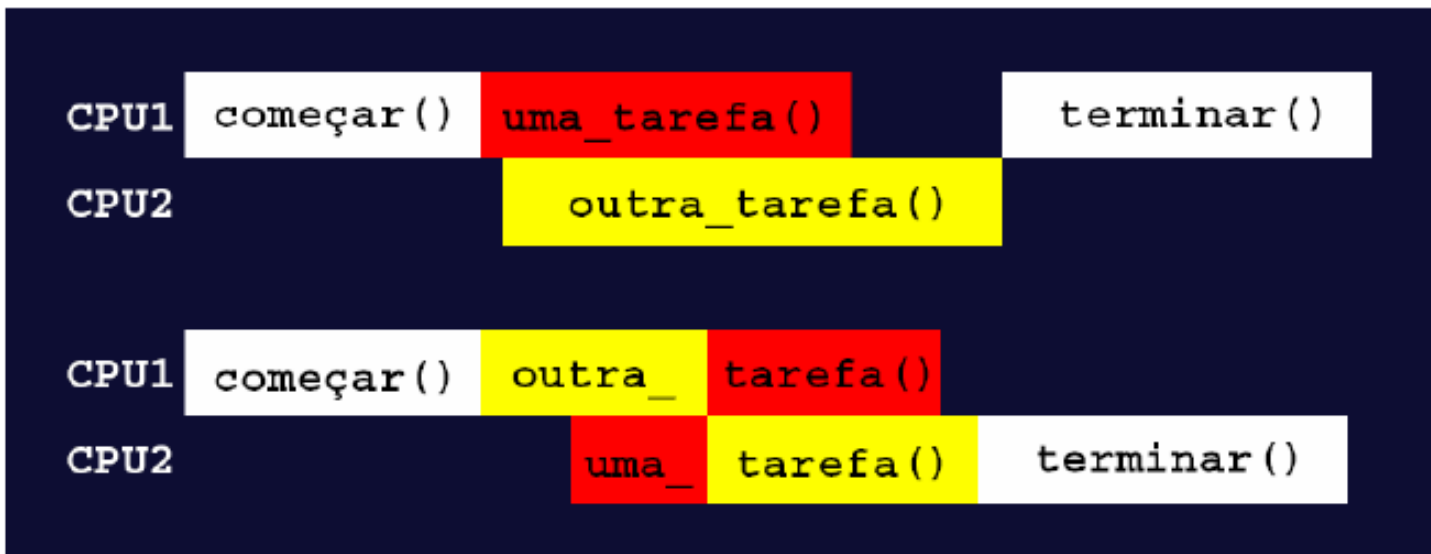
Concorrência ou Paralelismo Potencial

- Concorrência ou paralelismo potencial diz-se quando um programa possui tarefas que podem ser executadas em qualquer ordem sem alterar o resultado final.



Paralelismo

- Paralelismo diz-se quando as tarefas de um programa são executadas em simultâneo em mais do que um processador.



Paralelismo Implícito

- O paralelismo diz-se implícito quando cabe ao compilador e ao sistema de execução:
 - Detectar o paralelismo potencial do programa.
 - Atribuir as tarefas para execução em paralelo.
 - Controlar e sincronizar toda a execução.
- **Vantagens (+) e inconvenientes (-):**
 - (+) Liberta o programador dos detalhes da execução paralela.
 - (+) Solução mais geral e mais flexível.
 - (-) Difícil conseguir-se uma solução eficiente para todos os casos.

Paralelismo Explícito

- O paralelismo diz-se explícito quando cabe ao programador:
 - Anotar as tarefas para execução em paralelo.
 - Atribuir (possivelmente) as tarefas aos processadores.
 - Controlar a execução indicando os pontos de sincronização.
 - Conhecer a arquitetura dos computadores de forma a conseguir o máximo desempenho (aumentar localidade, diminuir comunicação, etc).
- **Vantagens (+) e inconvenientes (-):**
 - (+) Programadores experientes produzem soluções muito eficientes para problemas específicos.
 - (-) O programador é o responsável por todos os detalhes da execução (debugging pode ser deveras penoso).
 - (-) Pouco portátil entre diferentes arquiteturas.

Computação Paralela

- De uma forma simples, a computação paralela pode ser definida como o uso simultâneo de vários recursos computacionais de forma a reduzir o tempo necessário para resolver um determinado problema. Esses recursos computacionais podem incluir:
 - Um único computador com múltiplos processadores.
 - Um número arbitrário de computadores ligados por rede.
 - A combinação de ambos.

Principais benefícios da Programação Paralela

- **Melhor desempenho**
- Permite que várias tarefas sejam executadas simultaneamente em múltiplos núcleos de processamento, o que pode levar a uma execução mais rápida e eficiente do programa.

Principais benefícios da Programação Paralela

- **Maior escalabilidade**
- Ao dividir uma tarefa em várias tarefas menores que podem ser executadas em paralelo, a programação paralela pode aumentar a capacidade de processamento do sistema à medida que o número de núcleos de processamento é aumentado.

Principais benefícios da Programação Paralela

- Resposta mais rápida
- Em sistemas de tempo real, a programação paralela pode ser usada para garantir uma resposta mais rápida às entradas do usuário ou a outras entradas de dados em tempo real.

Principais benefícios da Programação Paralela

- **Melhor uso dos recursos**
- Ao utilizar vários núcleos de processamento, a programação paralela pode permitir um melhor uso dos recursos do sistema, reduzindo o tempo de ociosidade do processador.

Principais benefícios da Programação Paralela

- Solução de problemas mais rápida
- A programação paralela pode ser usada para executar várias simulações ou testes em paralelo, o que pode acelerar a solução de problemas em sistemas complexos.

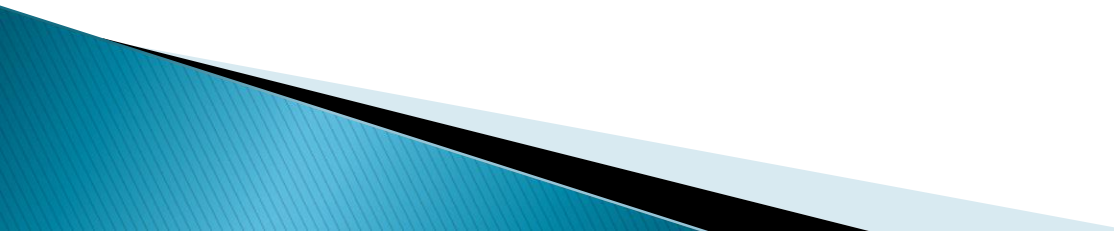
Principais desafios da Programação Paralela

- **Sincronização e comunicação**
- Ao executar várias tarefas em paralelo, é necessário garantir que as tarefas sejam executadas na ordem correta e que os dados sejam compartilhados e atualizados corretamente entre as diferentes tarefas.

Principais desafios da Programação Paralela

- **Escalonamento**
- É necessário garantir que as tarefas sejam distribuídas de maneira uniforme entre os vários núcleos de processamento disponíveis e que o tempo de execução seja minimizado.

Principais desafios da Programação Paralela

- Granularidade
 - É importante dividir as tarefas em pedaços pequenos o suficiente para permitir a execução paralela, mas grandes o suficiente para evitar sobrecarga com a comunicação e sincronização excessiva.
- 

Principais desafios da Programação Paralela

- **Paralelismo imperfeito**
- Nem todas as tarefas podem ser paralelizadas de forma eficiente e algumas partes do código podem limitar o paralelismo, o que pode levar a gargalos no desempenho.

Principais desafios da Programação Paralela

- **Debugging**
- A depuração de programas paralelos é mais difícil do que a depuração de programas sequenciais, uma vez que é necessário lidar com interações complexas entre os diferentes threads ou processos em execução simultânea.

Principais desafios da Programação Paralela

- **Overhead**
- A programação paralela pode exigir sobrecarga adicional, como a criação e destruição de threads, gerenciamento de memória compartilhada, etc., o que pode reduzir o desempenho em algumas situações.

Principais técnicas de programação paralela

- **Multithreading**
- Essa técnica envolve a divisão de uma tarefa em várias threads, que podem ser executadas simultaneamente em um ou mais núcleos de processamento. Os threads podem compartilhar recursos e comunicar-se entre si para concluir a tarefa.

Principais técnicas de programação paralela

- **Processamento em lotes (batch processing)**
- Essa técnica envolve a divisão de uma grande quantidade de dados em lotes menores, que podem ser processados em paralelo em diferentes núcleos de processamento. Cada núcleo processa um lote diferente, o que pode acelerar o processamento total dos dados.

Principais técnicas de programação paralela

- **Vetorização**
- Essa técnica envolve o uso de instruções SIMD (Single Instruction Multiple Data) para executar a mesma operação em vários elementos de dados ao mesmo tempo. Isso pode ser útil em operações matemáticas intensivas, como processamento de imagem ou vídeo.

Principais técnicas de programação paralela

- **Paralelismo de dados**
- Essa técnica envolve a divisão de um conjunto de dados em pedaços menores e processando cada pedaço em um núcleo de processamento diferente. Isso pode ser útil para operações que envolvem processamento de grandes conjuntos de dados, como cálculos científicos ou de simulação.

Principais técnicas de programação paralela

- **Paralelismo de tarefas (task parallelism)**
- Essa técnica envolve a divisão de uma tarefa em várias subtarefas menores, que podem ser executadas simultaneamente em diferentes núcleos de processamento. Isso pode ser útil em programas que envolvem várias etapas de processamento, como renderização de gráficos ou processamento de áudio.

Aplicativos do mundo real que se beneficiam da programação paralela

- **Simulações científicas**
- Muitos aplicativos de simulação científica, como modelagem climática, dinâmica de fluidos computacional e simulação de mecânica quântica, exigem um grande número de cálculos intensivos de processamento, o que pode ser acelerado pela programação paralela.

Aplicativos do mundo real que se beneficiam da programação paralela

- **Renderização de gráficos**
- A renderização de gráficos para jogos, animações e filmes requer o processamento de grandes quantidades de dados de imagem, o que pode ser acelerado pela programação paralela. Isso permite que as imagens sejam renderizadas em tempo real ou em prazos mais curtos.

Aplicativos do mundo real que se beneficiam da programação paralela

- **Análise de dados**
- Muitos aplicativos de análise de dados, como processamento de imagem, mineração de dados e aprendizado de máquina, envolvem o processamento de grandes conjuntos de dados, o que pode ser acelerado pela programação paralela.

Aplicativos do mundo real que se beneficiam da programação paralela

- **Computação em nuvem**
- A computação em nuvem é frequentemente usada para processamento de grandes volumes de dados ou cálculos intensivos de processamento. A programação paralela pode ser usada para acelerar esses processos, permitindo que as tarefas sejam concluídas mais rapidamente.

Aplicativos do mundo real que se beneficiam da programação paralela

- **Processamento de transações financeiras**
- Processamento de grandes volumes de transações financeiras é um exemplo de um aplicativo que se beneficia de programação paralela. O processamento de transações em paralelo pode acelerar o tempo de resposta e a eficiência do sistema.

Aplicativos do mundo real que se beneficiam da programação paralela

- **Computação científica**
- Além de simulações científicas, outras áreas da computação científica, como modelagem molecular, otimização de sistemas e análise de dados biológicos, podem se beneficiar da programação paralela para acelerar o processamento e melhorar a precisão dos resultados.

Aplicativos do mundo real que se beneficiam da programação paralela

- **Processamento de sinais**
- Muitos aplicativos de processamento de sinais, como reconhecimento de voz e imagem, processamento de áudio e vídeo, e análise de dados sísmicos, podem ser executados em paralelo para melhorar o desempenho e a precisão dos resultados.

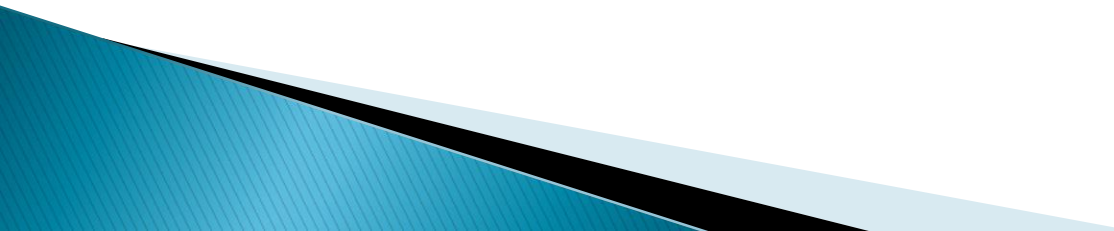
Aplicativos do mundo real que se beneficiam da programação paralela

- **Análise de risco**
- Muitas empresas usam a programação paralela para realizar análises de risco financeiro, modelagem de portfólio e simulações de monte carlo. Isso permite que as empresas tomem decisões mais informadas e precisas em tempo hábil.

Aplicativos do mundo real que se beneficiam da programação paralela

- Banco de dados
- Muitos sistemas de banco de dados distribuídos usam a programação paralela para acelerar o processamento de consultas e garantir a disponibilidade contínua do sistema.

Aplicativos do mundo real que se beneficiam da programação paralela

- Redes neurais
 - O treinamento de redes neurais profundas pode exigir enormes quantidades de cálculos. A programação paralela pode ser usada para acelerar esse processo, permitindo que as redes neurais sejam treinadas mais rapidamente e com maior precisão.
- 

Exemplo em Java usando a API de concorrência "java.util.concurrent"

```
1 import java.util.concurrent.ExecutorService;
2 import java.util.concurrent.Executors;
3 import java.util.concurrent.TimeUnit;
4
5 public class ParallelProcessingExample {
6     public static void main(String[] args) {
7         // conjunto de dados para serem processados
8         int[] data = new int[] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
9
10        // número de threads usadas para processamento paralelo
11        int numThreads = Runtime.getRuntime().availableProcessors();
12
13        // criação de pool de threads para processamento em paralelo
14        ExecutorService executor = Executors.newFixedThreadPool(numThreads);
15
16        // processamento de dados em paralelo usando threads
17        for (int i = 0; i < data.length; i++) {
18            final int index = i;
19            executor.execute(new Runnable() {
20                public void run() {
21                    // processamento de dados aqui
22                    System.out.println("Processed " + data[index] + " on thread " + Thread.currentThread().getName());
23                }
24            });
25        }
26
27        // aguarda o término do processamento de todas as threads
28        executor.shutdown();
29        try {
30            executor.awaitTermination(Long.MAX_VALUE, TimeUnit.NANOSECONDS);
31        } catch (InterruptedException e) {
32            e.printStackTrace();
33        }
34    }
35 }
```

<https://www.jdoodle.com/online-java-compiler/>

Código para copiar

```
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.TimeUnit;

public class ParallelProcessingExample {
    public static void main(String[] args) {
        // conjunto de dados para serem processados
        int[] data = new int[] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

        // número de threads usadas para processamento paralelo
        int numThreads = Runtime.getRuntime().availableProcessors();

        // criação de pool de threads para processamento em paralelo
        ExecutorService executor = Executors.newFixedThreadPool(numThreads);

        // processamento de dados em paralelo usando threads
        for (int i = 0; i < data.length; i++) {
            final int index = i;
            executor.execute(new Runnable() {
                public void run() {
                    // processamento de dados aqui
                    System.out.println("Processed " + data[index] + " on thread " + Thread.currentThread().getName());
                }
            });
        }

        // aguarda o término do processamento de todas as threads
        executor.shutdown();
        try {
            executor.awaitTermination(Long.MAX_VALUE, TimeUnit.NANOSECONDS);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

Explicações do código

- ▶ A API "java.util.concurrent" é usada para criar um pool de threads para processar um conjunto de dados em paralelo. O número de threads é definido com base no número de núcleos de CPU disponíveis no sistema.
- ▶ Um loop é executado para processar cada elemento do conjunto de dados em paralelo usando threads. Cada thread executa a função "run" para processar um único elemento do conjunto de dados.

Explicações do código

- ▶ Por fim, o código aguarda o término do processamento de todas as threads no pool antes de finalizar.
- ▶ Há outras APIs e bibliotecas disponíveis em Java para programação paralela, mas o conceito geral é semelhante, dividir uma tarefa em partes menores para serem executadas simultaneamente em vários núcleos de CPU ou processadores para melhorar o desempenho e a eficiência.

Referências desta Aula

- DE ROSE, César A. F.; NAVAUX, Philippe O. A. Arquiteturas Paralelas. Porto Alegre: Sagra-Luzzato, 2008.
- Aula montada com base no material do professor Ricardo Rocha – DCC-FCUP. Disponível em:
<https://www.dcc.fc.up.pt/~ricroc/aulas/0708/ppd/apontamentos/fundamentos.pdf>

FIM
Obrigado!

Rodrigo