DISCIPLINA

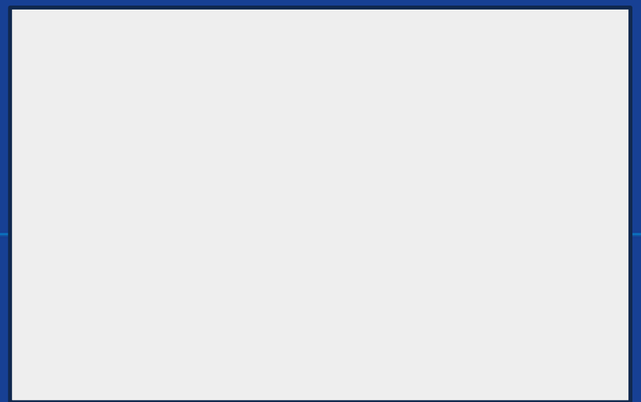
Introdução à Computação Paralela - Introdução ao Paralelismo -

Prof. Kayo Gonçalves

BACHARELADO EM TECNOLOGIA DA INFORMAÇÃO

UFRN | IMD

Adaptado do livro “An Introduction to Parallel Programming” do Peter Pacheco

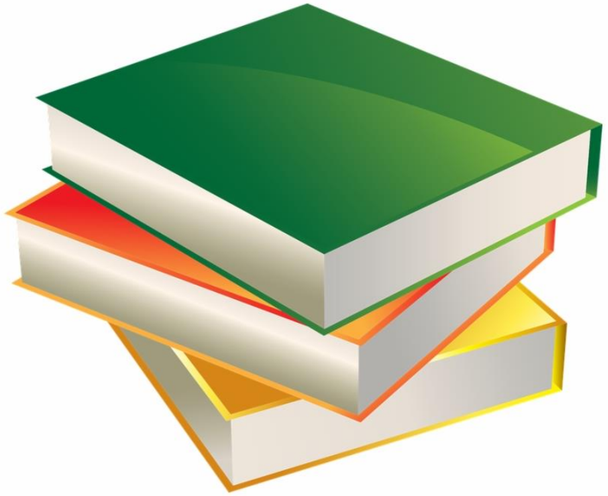


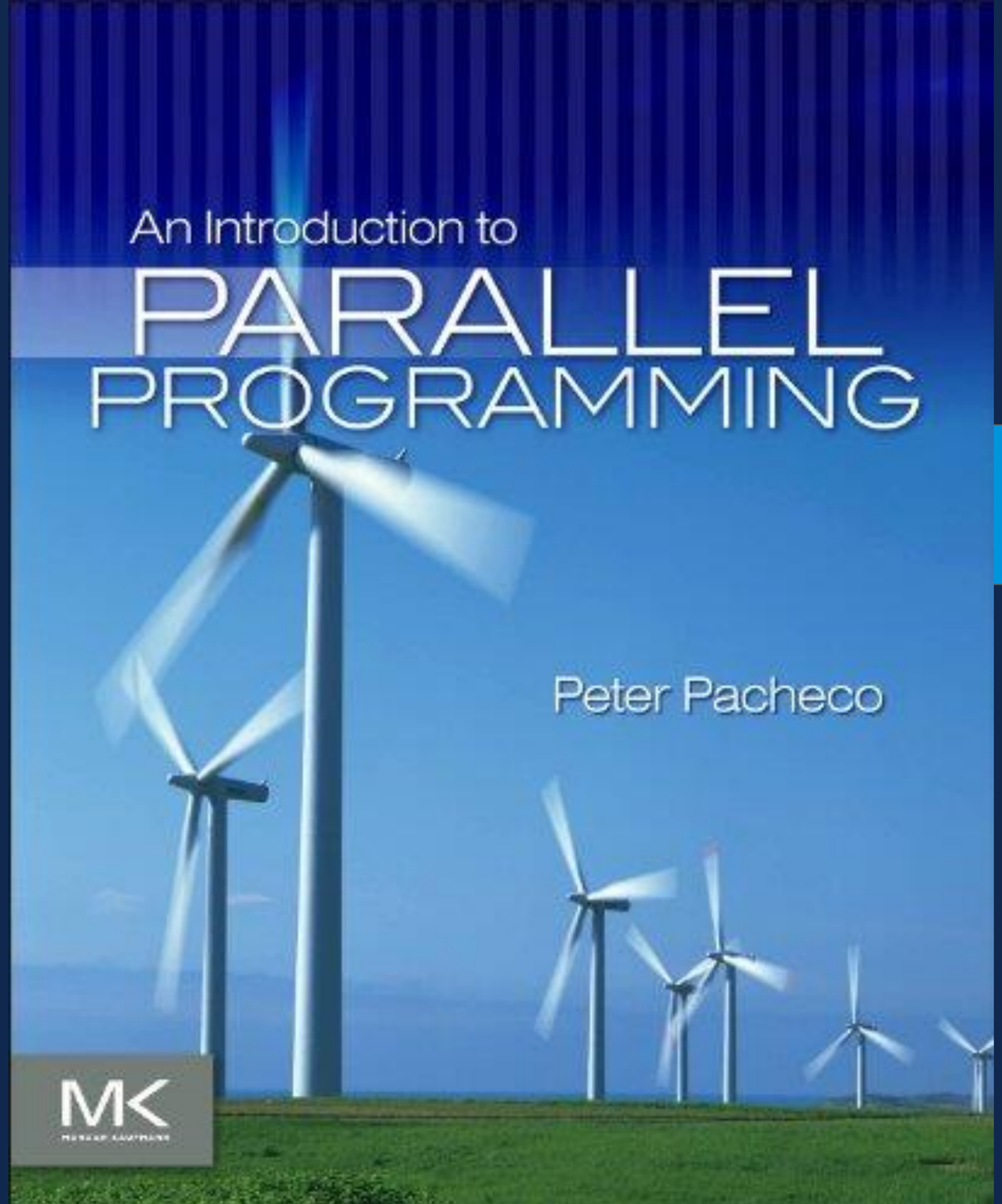
O que faremos

- Aprender a escrever programas explicitamente paralelos

- Usando a linguagem C/C++

- Usando três extensões diferentes para C/C++
    - Interface de transmissão de mensagens (MPI)
    - Linhas Posix (Pthreads)
    - OpenMP

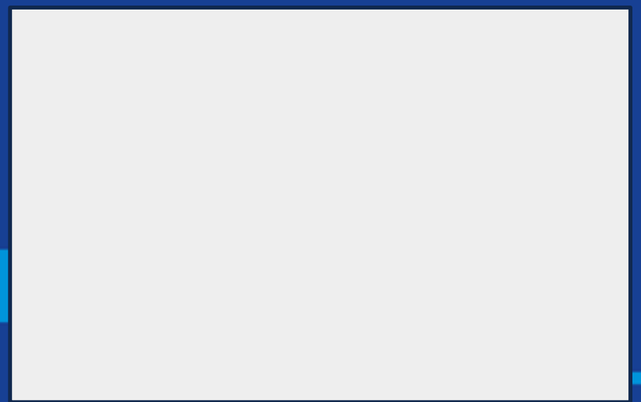




Bibliografia Base

An Introduction to Parallel Programming
Peter Pacheco

Contexto Histórico e Motivações



Mudança dos tempos

- De 1986 – 2002, microprocessadores aumentavam o desempenho numa média de 50% por ano.

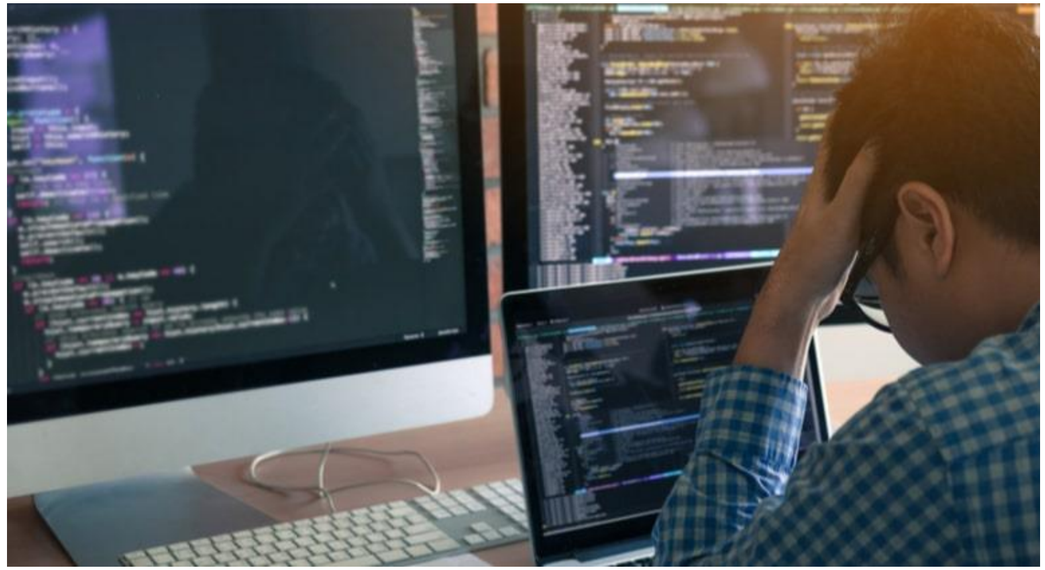
- Desde então, caiu para 20% por ano.

- Ler: Lei de Moore

Agora é com programadores

- Adicionar novos processadores não ajuda muito se os programadores não estão ciente deles.

… ou não sabem como utilizá-los.
Programas Seriais não se beneficiam desta abordagem.



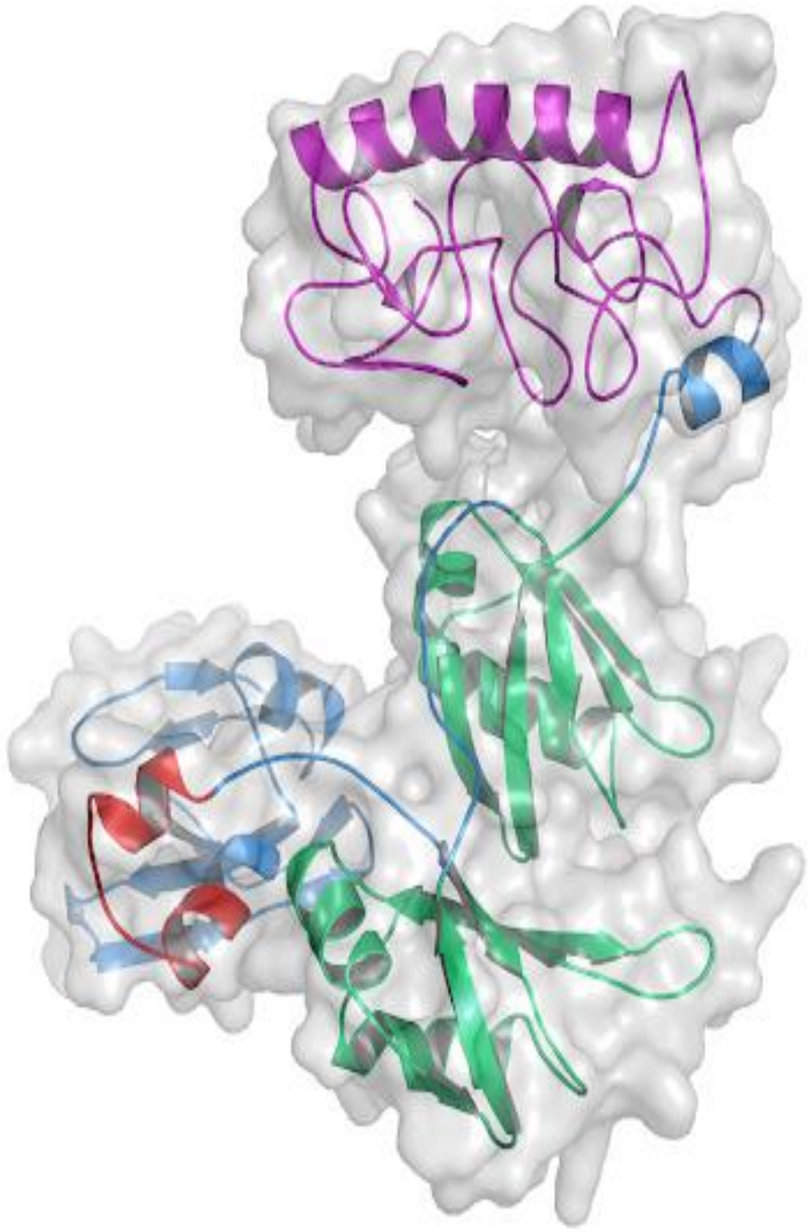
Por que precisamos de desempenho

- O poder computacional é crescente, mas também são nossos problemas e necessidades

Problemas que nunca sonhamos de serem resolvidos no passado, como decodificar o genoma humano.

Problemas com mais complexidade estão ainda para serem resolvidos.

Modelagem climática



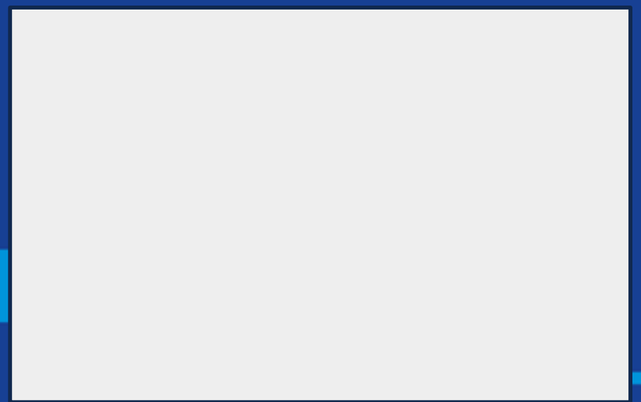
Dobramento de proteínas

Drogas medicinais

Energias

Análise de dados

O porquê de sistemas paralelos



Por que estamos construindo sistemas paralelos?

- Até agora, os aumentos de desempenhos foram atribuídos ao aumento do número de transistores

- Mas existem problemas inerentes

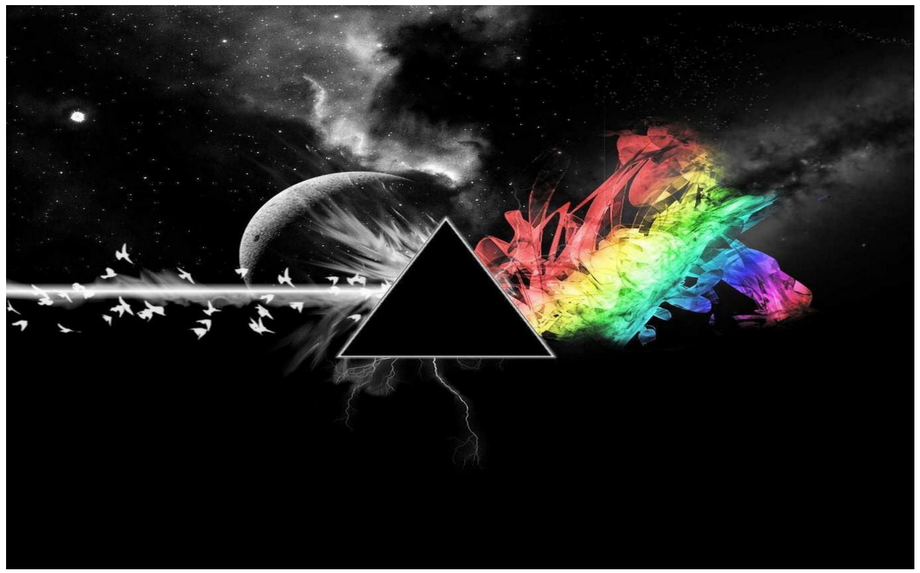
Uma pequena lição de física

- Menores transistores = processadores mais rápidos.

- Processadores mais rápidos = aumento do **consume energético**

- Aumento do consume energético = aumento do **aquecimento**

- Aumento do aquecimento = processadores **não confiáveis** e **dificuldade de resfriamento**



SOLUÇÃO: de “single-core” para “multi-core”

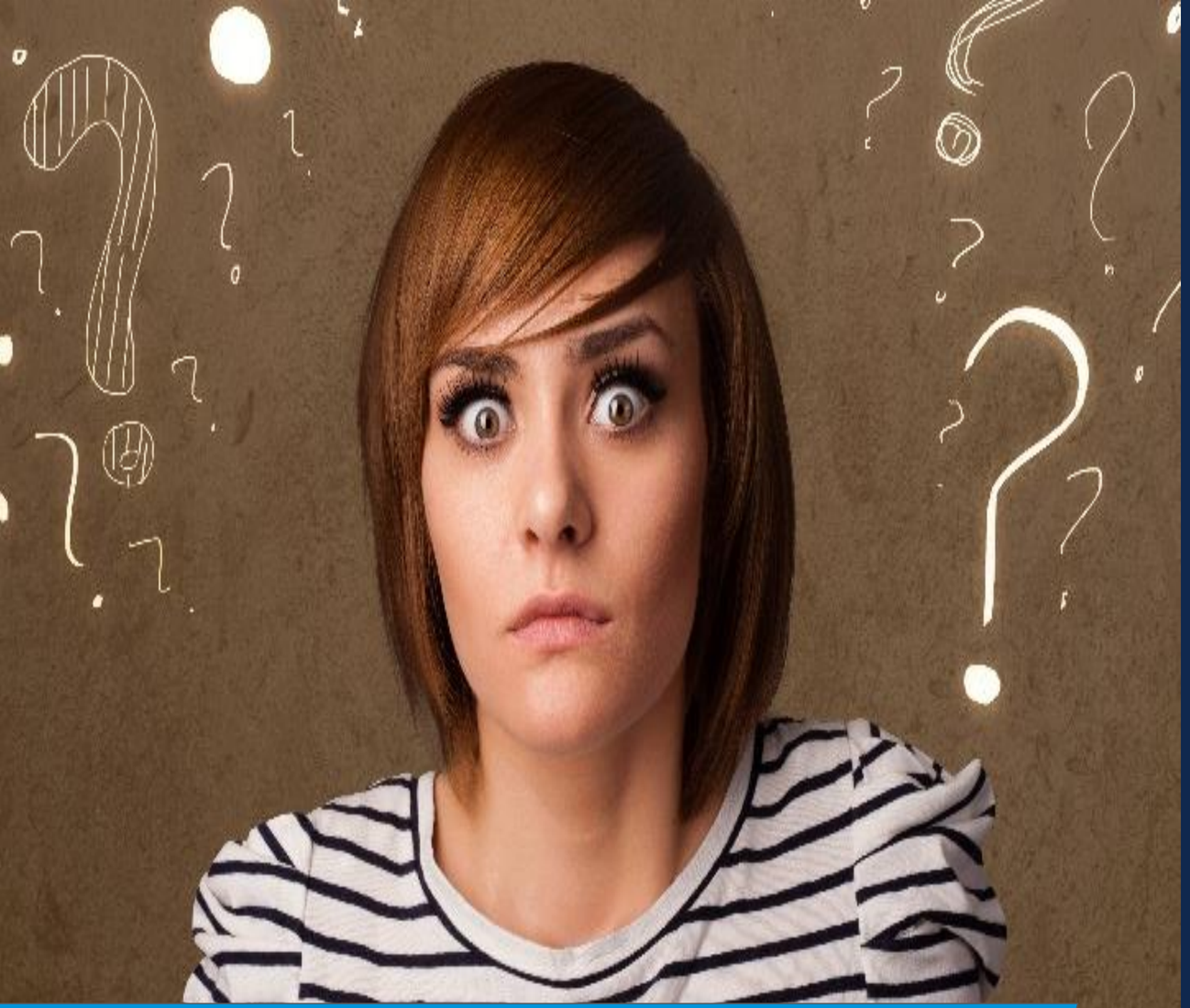
Introdução do paralelismo.

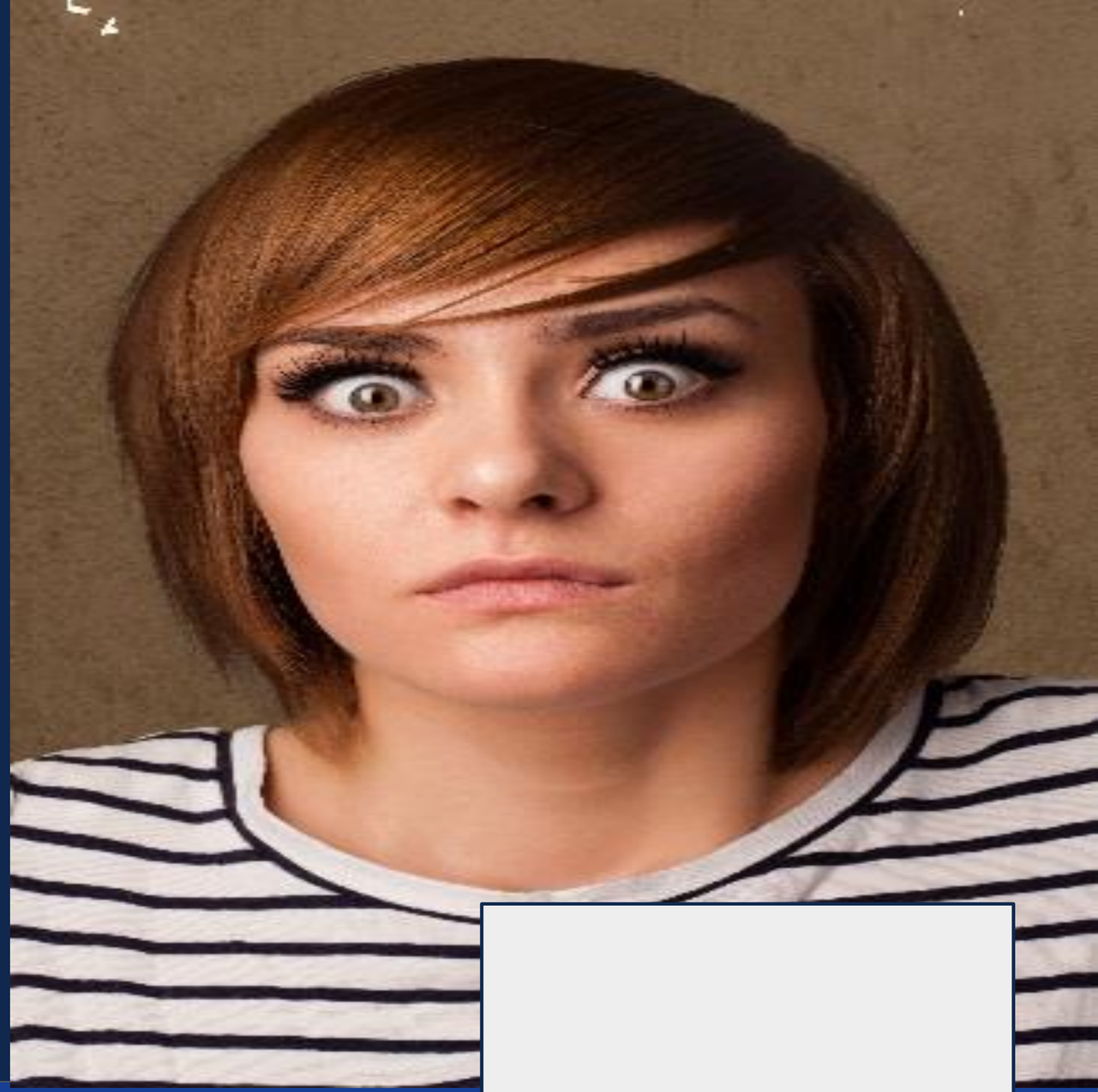
Por que precisamos escrever programas paralelos?

- Executar **múltiplas instâncias** de um código serial frequentemente NÃO é útil.

Pense em rodar múltiplas instâncias do seu jogo favorito. =D

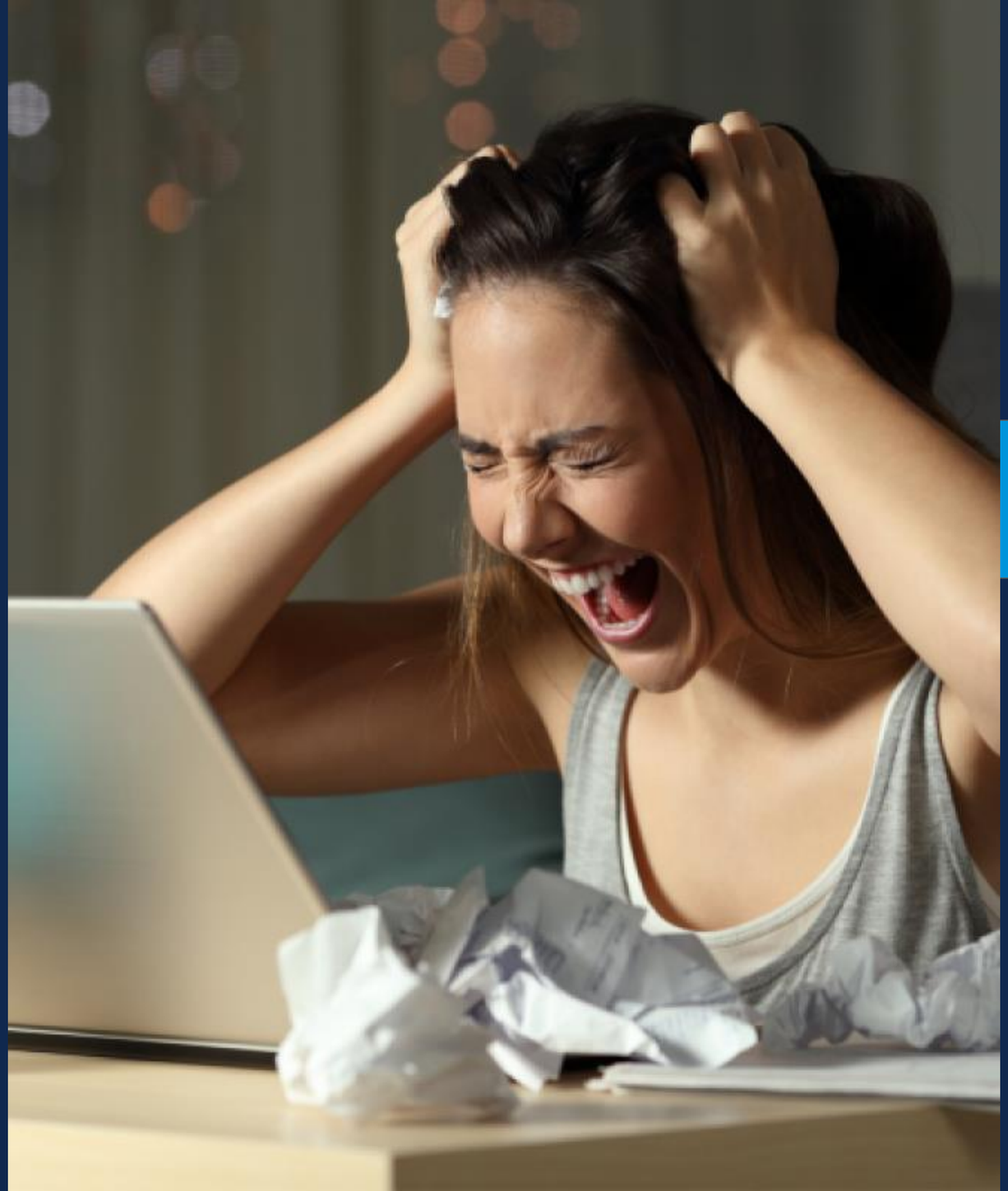
- O que realmente queremos é “rodar” **MAIS RÁPIDO**





Conversão automática serial-paralelo

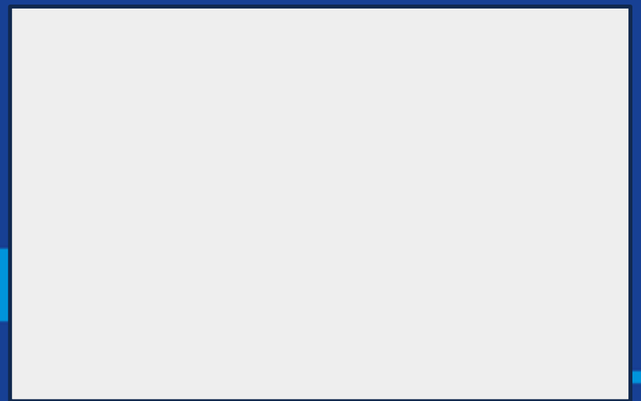
Programas **não são eficientes** na conversão automática de serial para paralelo



Solução!

Às vezes, a melhor solução paralela é recuar e **criar um algoritmo inteiramente novo**

Exemplo eficaz e ineficaz



Exemplo 1

Compute **n valores** de um vetor e some-os.
Solução serial:

```
sum = 0;
for (i = 0; i < n; i++) {
    x = Compute_next_value(. . .);
    sum += x;
}
```

Exemplo 2

Compute **n valores** e some-os. Solução paralela:

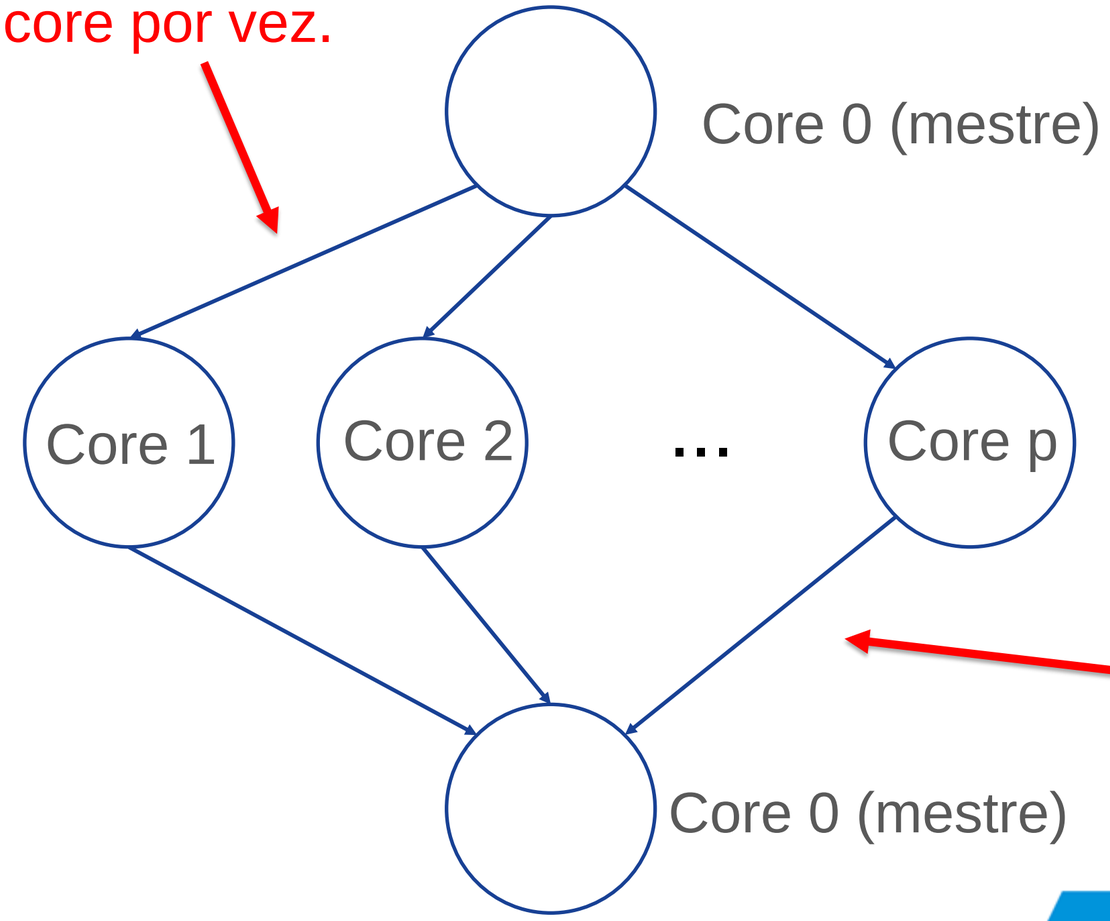
ALGORITMO:
- Um core (mestre) envia **n/p números** diferentes para cada um dos demais **p cores** (escravos)

- Cada um dos ***p cores*** soma **n/p números**

- Cada **core** envia o resultado para o **mestre** e este acumula os valores recebidos.

Exemplo 2 (continução)

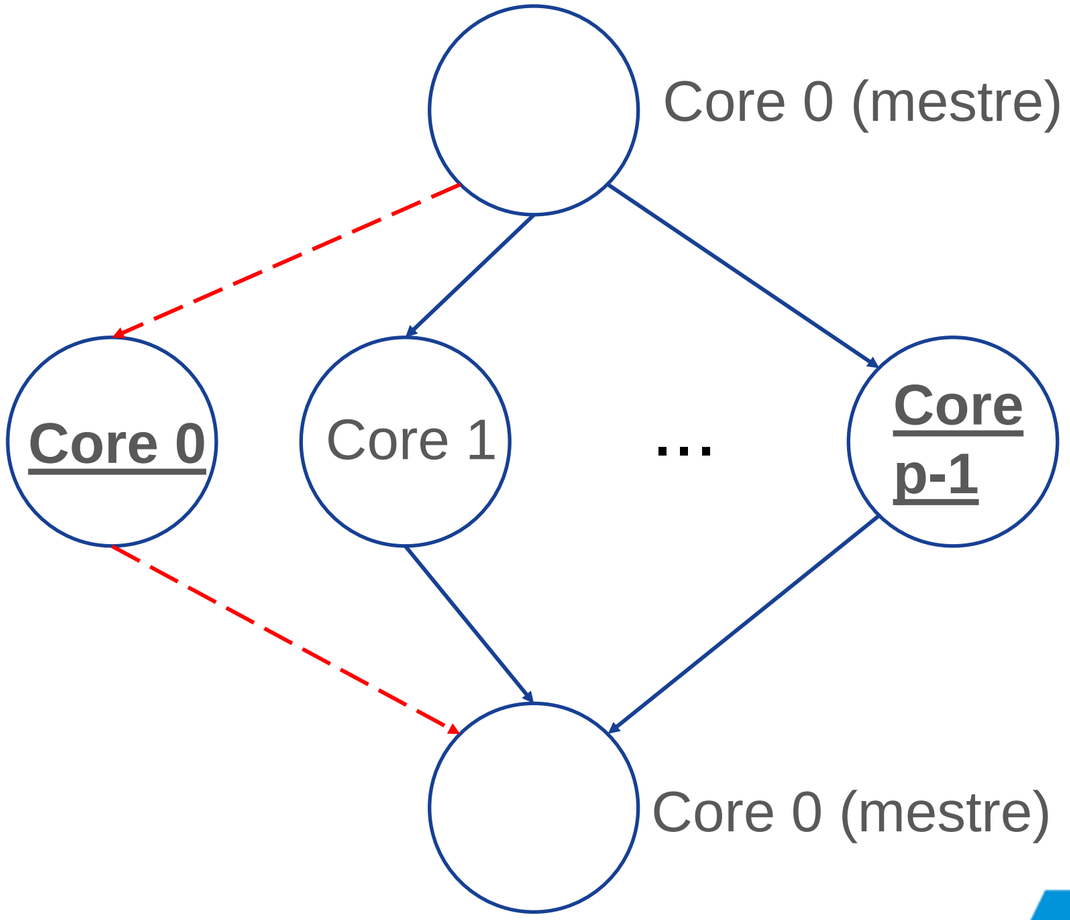
O envio ocorre de
1 core por vez.



A recepção ocorre
de 1 core por vez.

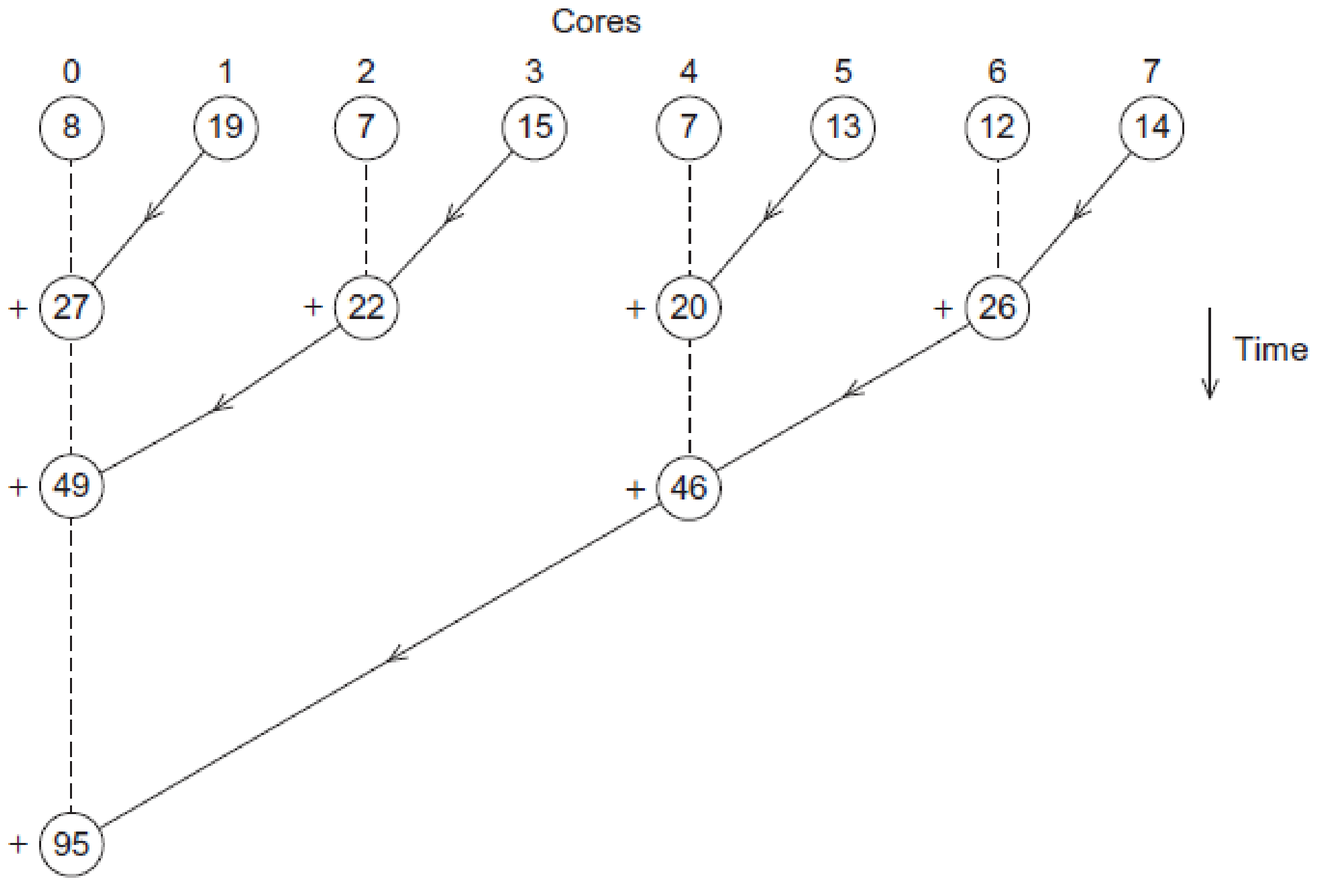
Exemplo 3

Melhorando o código!!!



Exemplo 4

Melhorando o código AINDA MAIS!!!



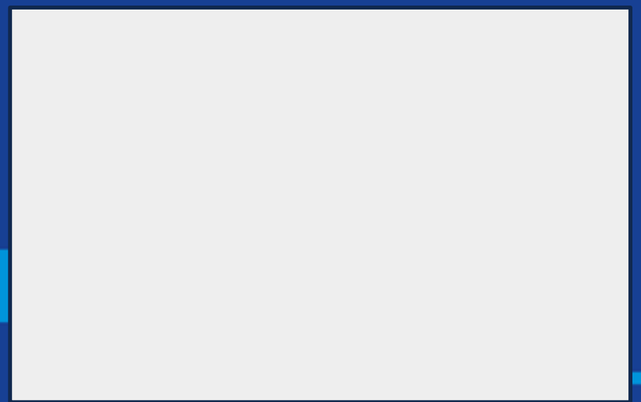
Análise 1

- Considerando 1 mestre e 7 escravos:
  - No exemplo 3, o mestre executa 7 envios, 7 recebimentos e 7 adições.
  - No exemplo 4, o mestre executa 7 envios, 3 recebimentos e 3 adições.
- **A diferença aumenta com maior número de cores**

Análise 2

- Considerando 1 mestre e 999 escravos:
  - No exemplo 3, o mestre executa 999 envios, 999 recebimentos e 999 adições.
  - No exemplo 4, o mestre executa 999 envios, **10 recebimentos e 10 adições.**

Paralelismo de tarefa e de dados



Como escrever algoritmos paralelos?

- Paralelismo de tarefa (task)
  - Cada core realiza tarefa diferente em todos os dados.
- Paralelismo de dados (data)
  - Cada core realiza tarefa semelhante em partes diferentes dos dados.

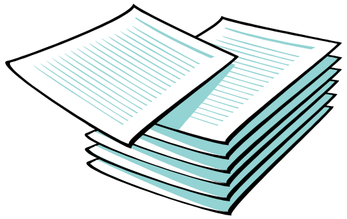
Exemplo 1:

Correção de 300 provas com 15 questões cada.

Assistentes A1, A2 e A3

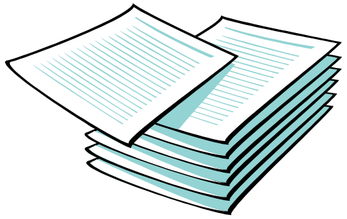
A felicidade no olhar de quem vai passar o dia corrigindo prova

A1



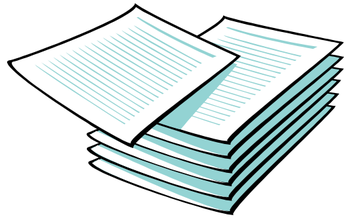
100 provas

A2



100 provas

A3

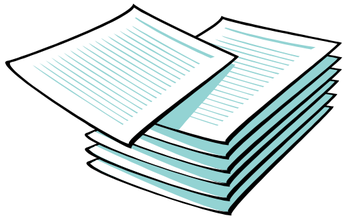


100 provas

Paralelismo de dados

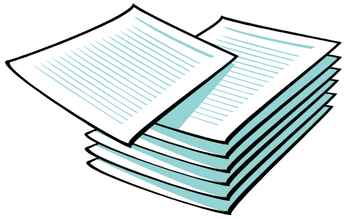
É muito trabalho

A1



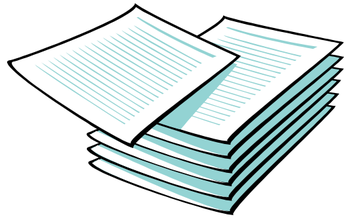
Questões 1-5

A2



Questões 11-15

A3



Questões 6-10

Paralelismo de tarefas

Continua sendo muito trabalho

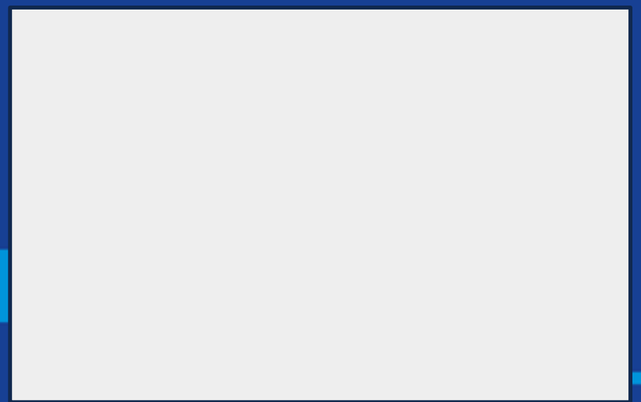
Coordenação

- **Comunicação** - um ou mais núcleos enviam suas somas parciais atuais para outro núcleo.

- **Balanceamento de carga** - compartilhe o trabalho de maneira uniforme entre os núcleos, para que não haja muita carga.

- **Sincronização** – Quando um processador para a sua execução porque depende de outro(s) para continuar.

Tipos de sistemas paralelos



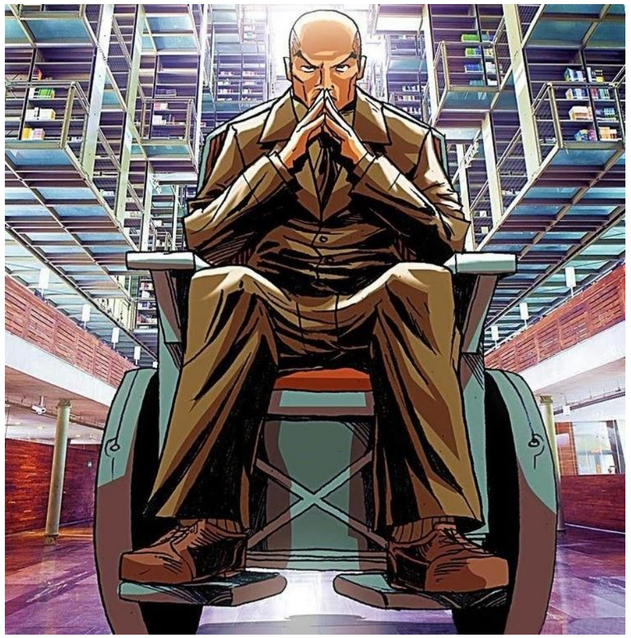
Tipos de Sistemas Paralelos

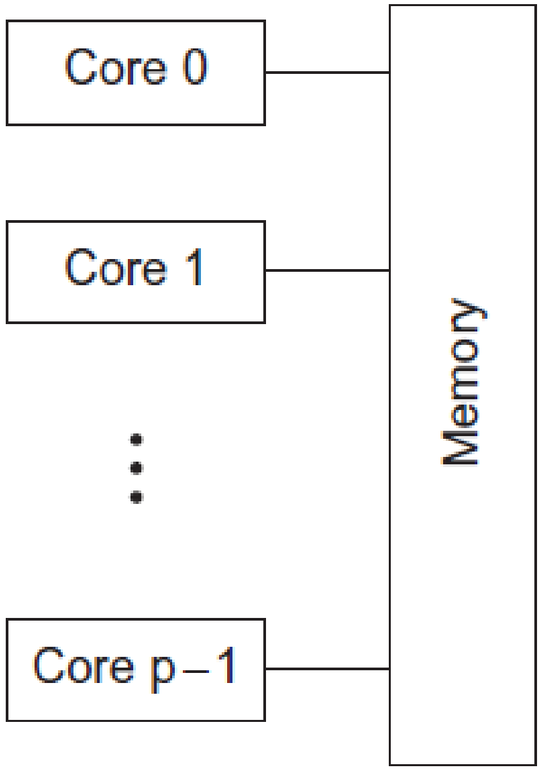
Memória compartilhada

- Os cores **compartilham o acesso** à memória do computador.
- **Coordene** os cores solicitando que examinem e atualizem os locais de memória compartilhada.

Memória distribuída

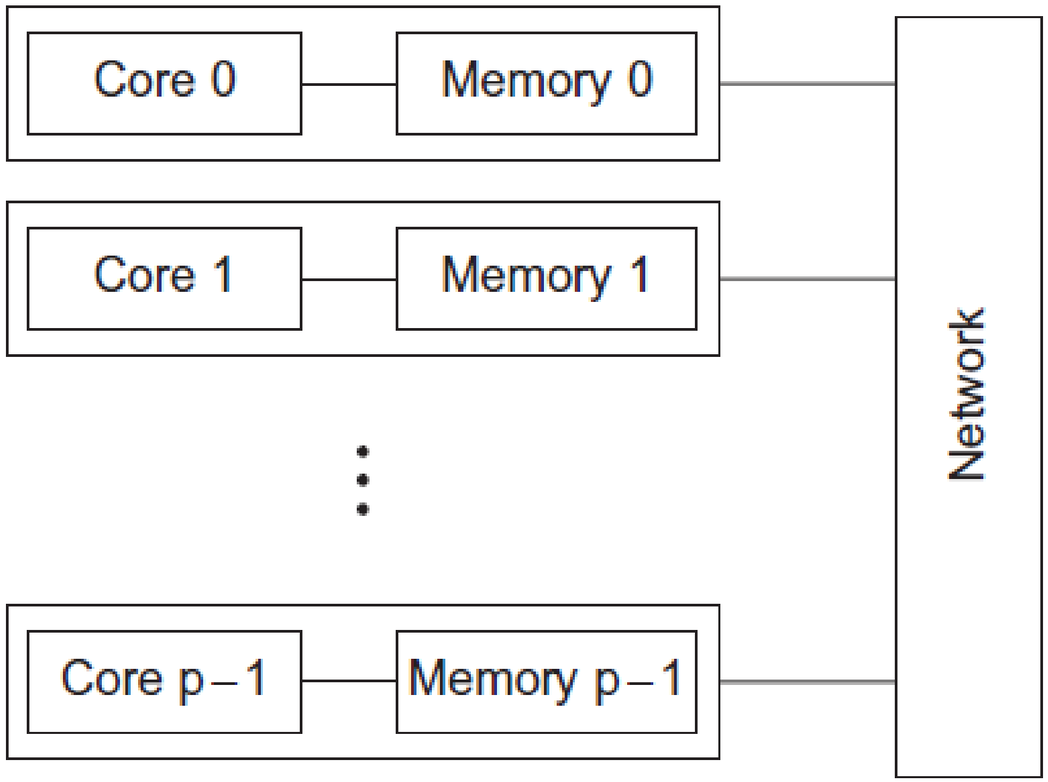
- Cada core tem **sua própria** memória privada.
- Os cores devem se **comunicar explicitamente** enviando mensagens pela rede.





(a)

Memória Compartilhada



(b)

Memória Distribuída

UFRN | IMD