universidade federal de pelotas
CDTec
centro de desenvolvimento tecnológico

computação
Universidade Federal de Pelotas

IPPD
Hoje: Introdução ao Processamento Paralelo

Prof. Dr. Rafael P. Torchelsen
rafael.torchelsen@inf.ufpel.edu.br

Visão Geral

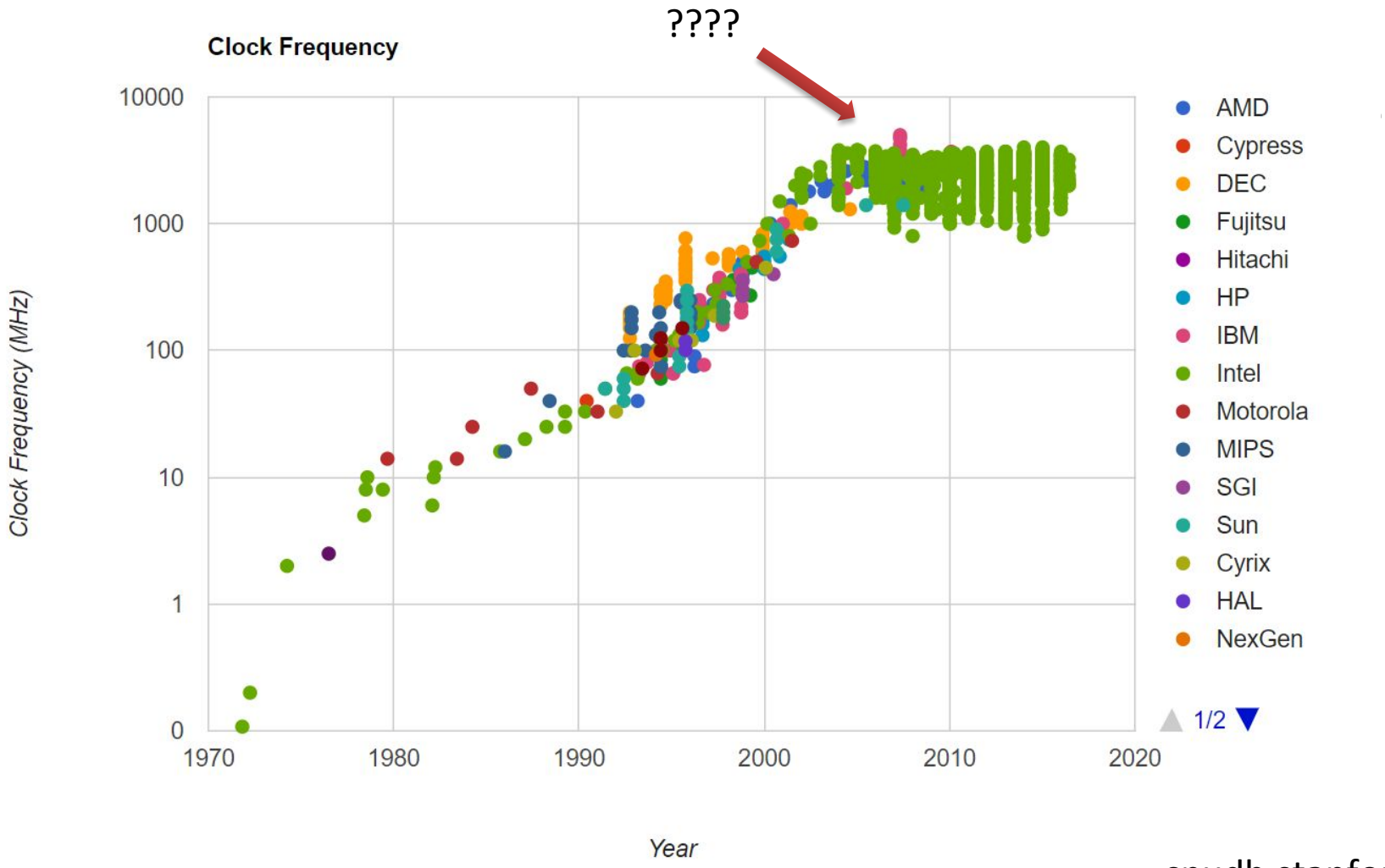
computação
Universidade Federal de Pelotas

- Por que sempre precisamos de mais desempenho?
- Por que construímos sistemas paralelos?
- Por que precisamos construir programa paralelos?
- Como escrever programas paralelos?
- Qual a diferenças entre sistemas paralelos?

Evolução do desempenho

computação
al de Pelotas

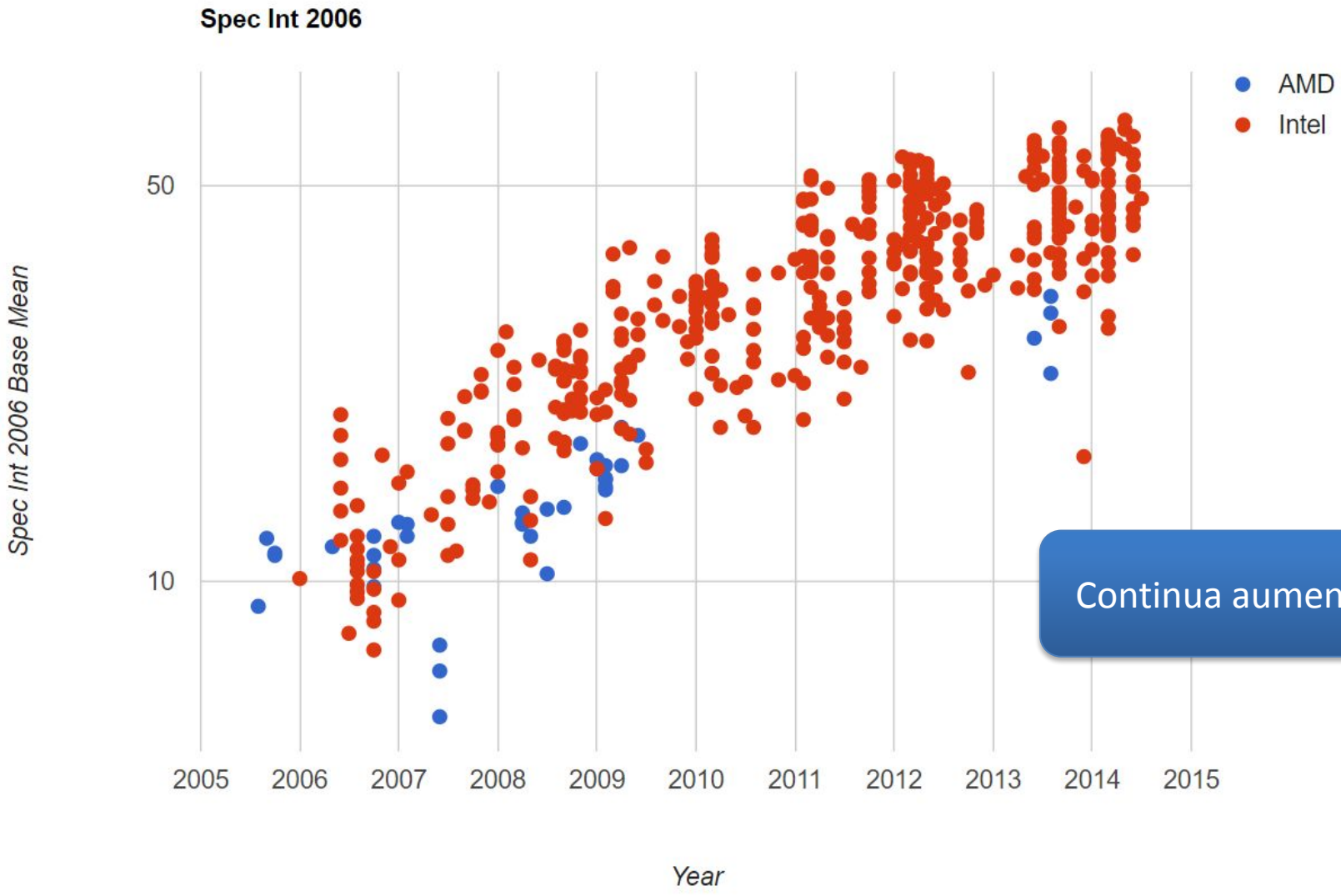
Clock Frequency



cpudb.stanford.edu

Desempenho de um único core

computação
Universidade Federal de Pelotas

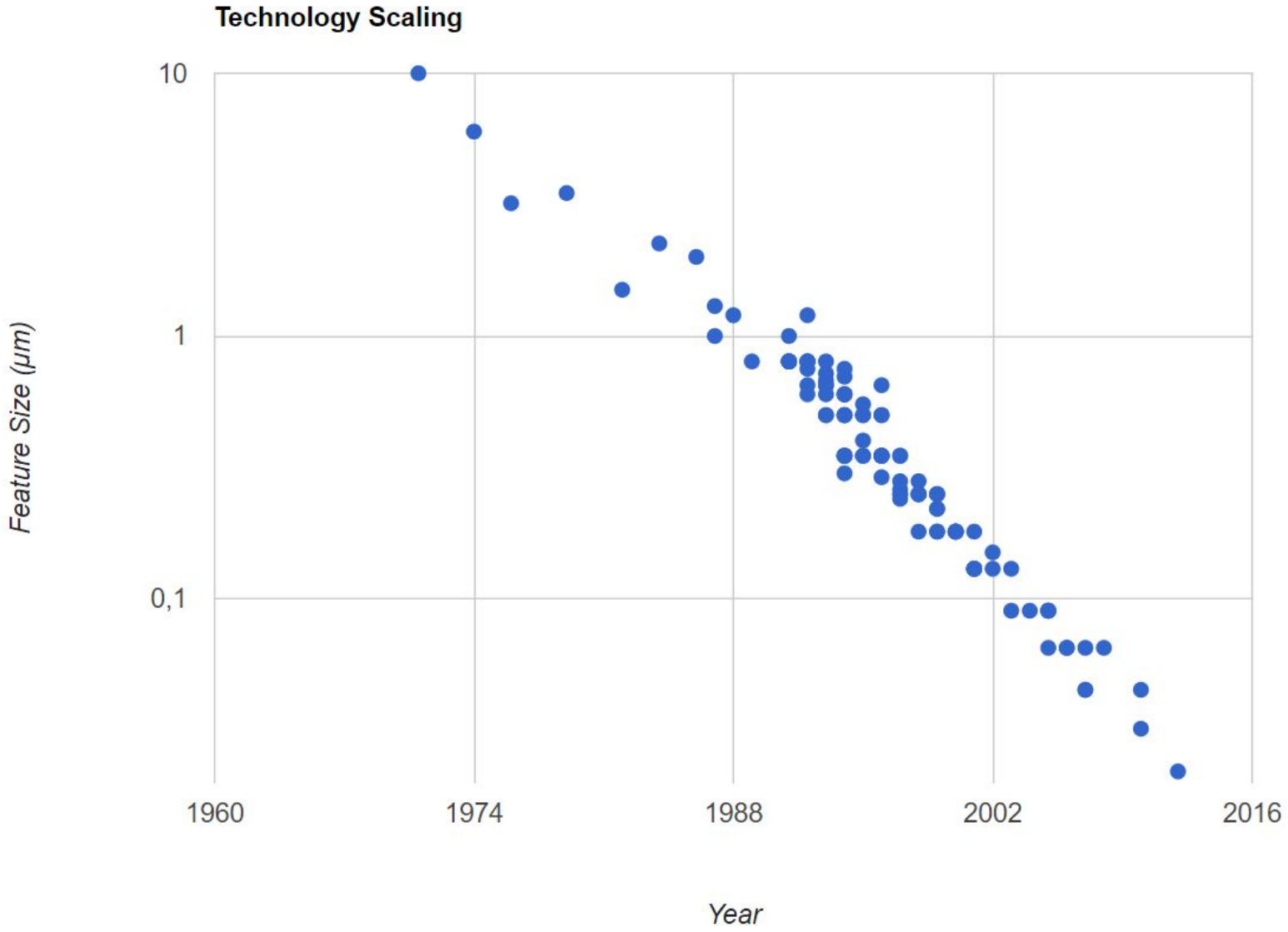


Continua aumentando...

cpudb.stanford.edu

Limite?

computação
Universidade Federal de Pelotas

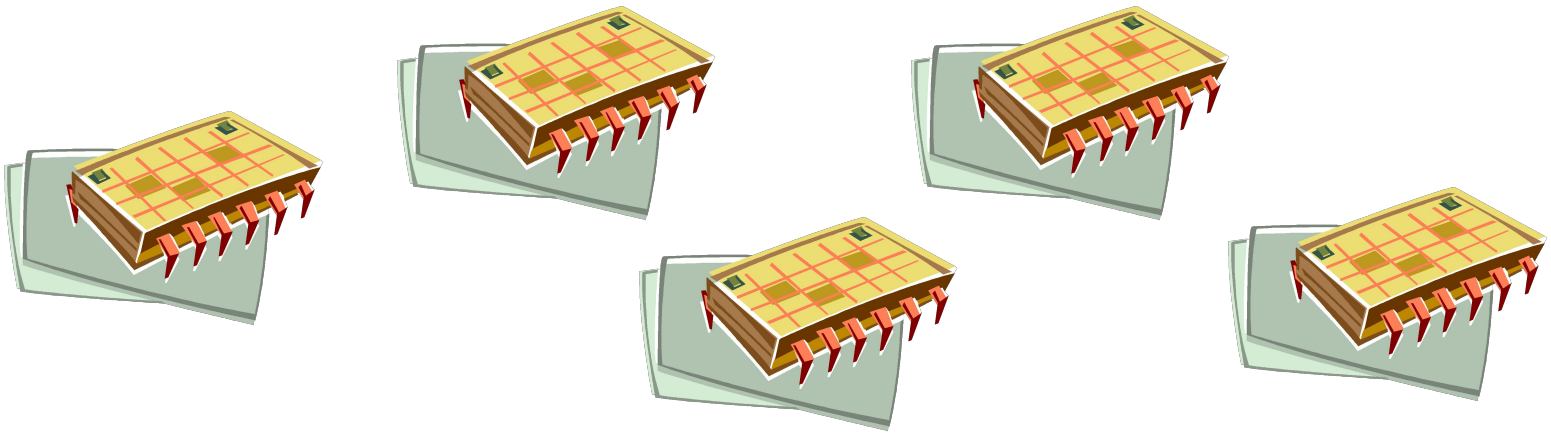


cpudb.stanford.edu

Solução?

computação
Universidade Federal de Pelotas

- Ao invés de construir processadores mais rápidos, colocar vários na mesma placa: **processamento paralelo**



Passamos o problema para onde?

computação
Universidade Federal de Pelotas

Programadores estão com o token

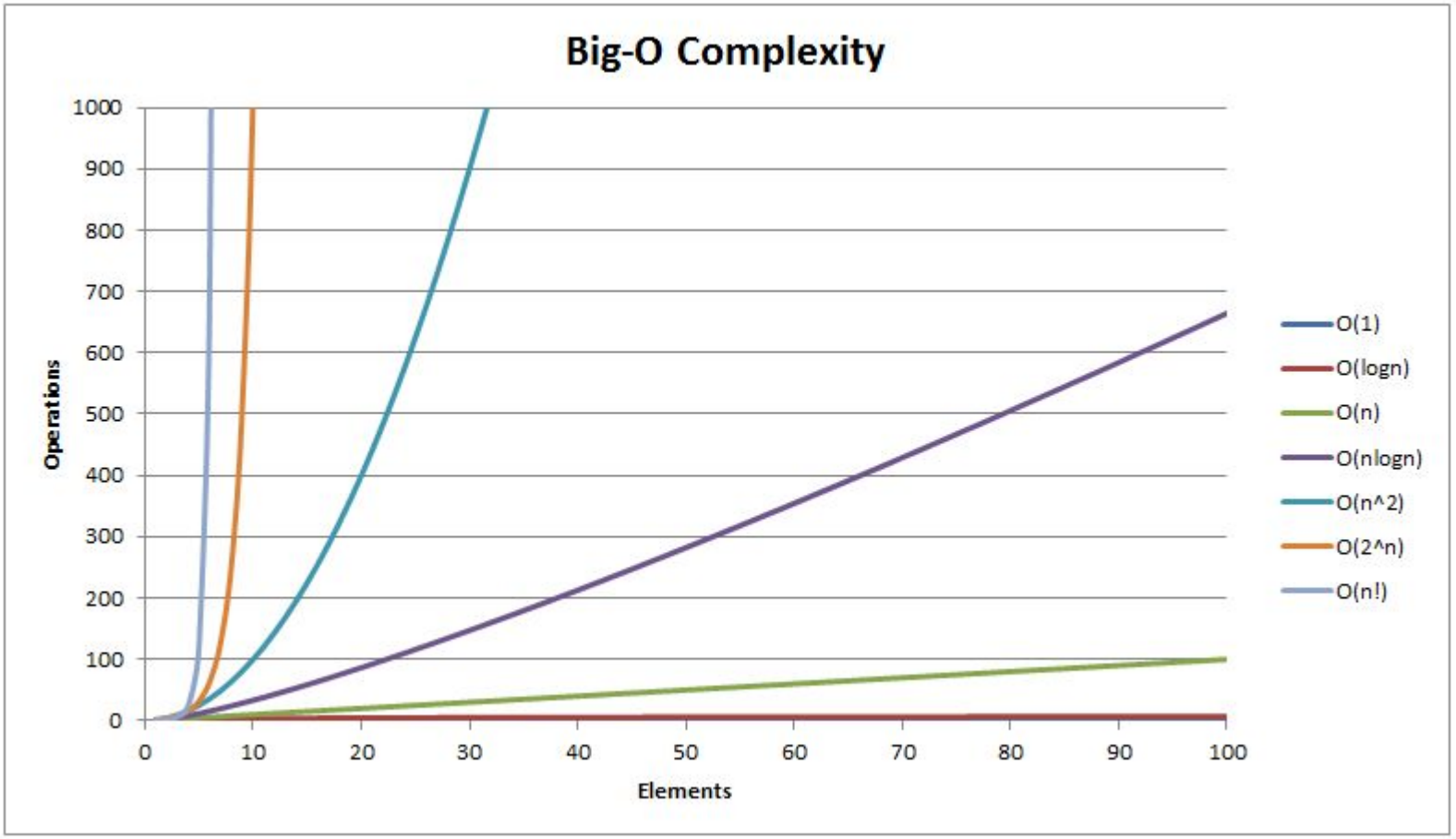
computação
Universidade Federal de Pelotas

- Não adianta ter vários processadores se os programadores não os usam...
- ... Ou pior, não sabem usá-los.

- Programas seriais, na maioria das vezes, não se beneficiam do atual aumento de desempenho.
  - Lembram dos gráficos de evolução?

Por que precisamos de mais desempenho?

computação
Universidade Federal de Pelotas



+ desempenho = soluções

computação
Universidade Federal de Pelotas

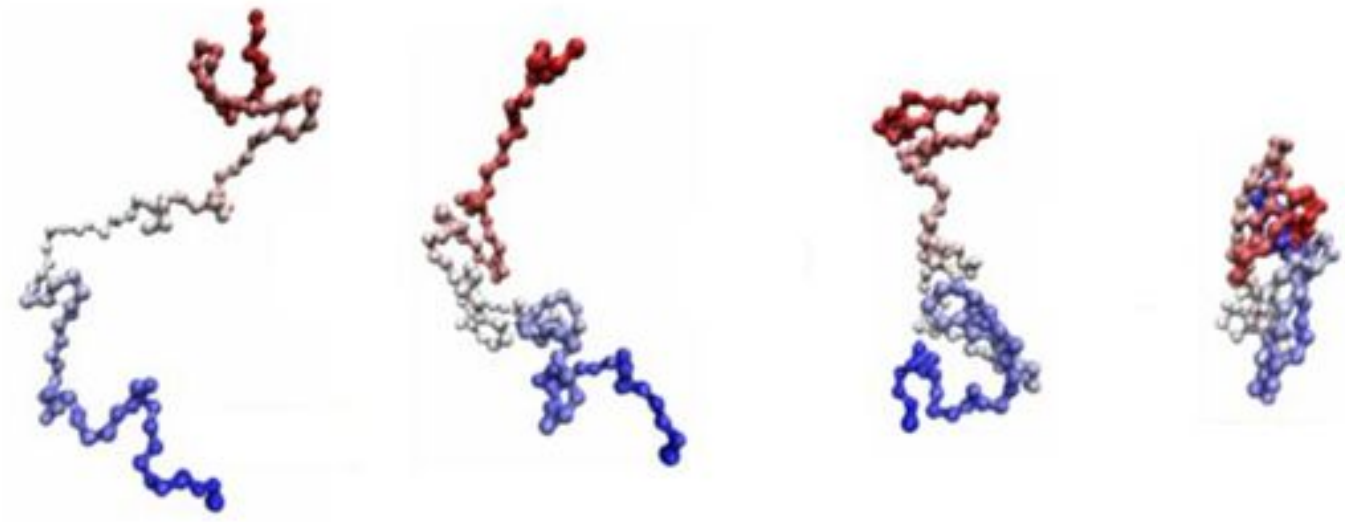
Problemas que pensávamos impossíveis de solucionar na prática agora são resolvíveis

Previsão do tempo

computação
Universidade Federal de Pelotas

Decodificação de proteínas

computação
Universidade Federal de Pelotas



Remédios

computação
Universidade Federal de Pelotas

Energia

computação
Universidade Federal de Pelotas

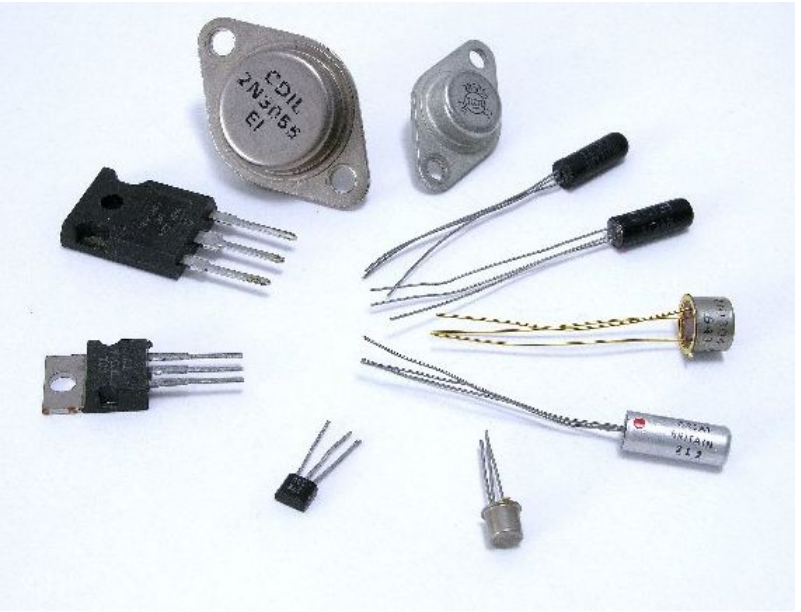
Análise de dados

computação
Universidade Federal de Pelotas

Por que sistemas paralelos são a solução?

computação
Universidade Federal de Pelotas

- Até agora o desempenho aumentava em relação ao aumento de densidade de transistores...
- ... mas existem problemas.



A física está no caminho

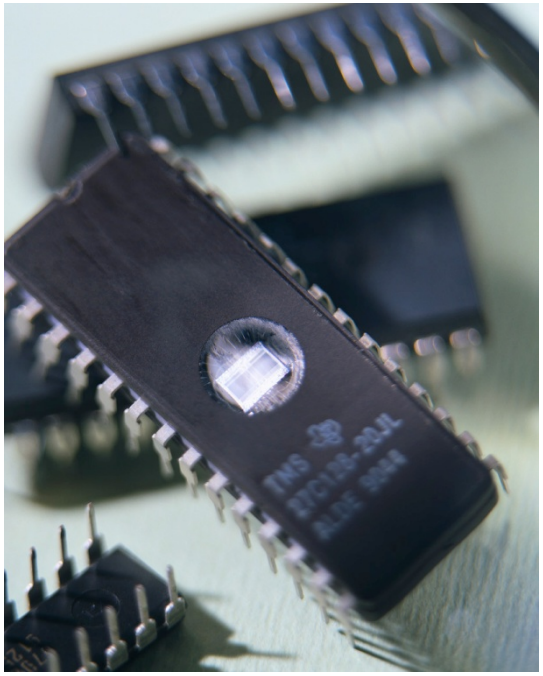
computação
Universidade Federal de Pelotas

- Transistores menores = processadores mais rápidos.
- Processadores mais rápidos = aumento no consumo de energia.
- Aumento no consumo de energia = aumento no calor.
- Aumento de calor = processadores instáveis.

Solução

computação
Universidade Federal de Pelotas

- Passar de um único processador para vários
  - Multi-core
- “core” = Unidade central de processamento (CPU)



Processamento Paralelo

Por que precisamos escrever programa paralelos?

- Rodar várias instâncias do mesmo programa geralmente é inútil.
  - Por exemplo, por que ordenar o mesmo conjunto de números várias vezes ao mesmo tempo?

- O que queremos é ordenar mais rapidamente.

Outros motivos

computação
Universidade Federal de Pelotas

- Dividir recursos, memória, cores em outros computadores, etc.
- Usar múltiplos processadores baratos e lentos para se ter um computador de alto desempenho.
- Ser escalável!

Precisamos reescrever os programas

computação
Universidade Federal de Pelotas

- Precisamos converter soluções seriais em paralela
  - Muitas vezes é simples, basta o programador ser bom (o que não é comum).
  - Porém, nem sempre é simples, por exemplo, problemas naturalmente seriais ou pouco paralelizáveis.
- Converter automaticamente de serial para paralelo?
  - Difícil
  - Quando possível o resultado é limitado

Conversão automática

computação
Universidade Federal de Pelotas

- Algumas estruturas paralelas podem ser identificadas no código e convertidas automaticamente para paralela.
- Porém, geralmente isso não ajuda muito.
- O melhor é gerar uma nova solução com paralelismo em mente do que converter pedaços de um programa para paralelo.

Exemplo

- Computar n valores e soma-los
- Solução serial:

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

Exemplo (cont.)

computação
Universidade Federal de Pelotas

- Com múltiplos cores (p), onde p é menor que n
- Cada core faz uma soma parcial de algo próximo a n/p

Granularidade

```
my_sum = 0;
my_first_i = . . . ;
my_last_i = . . . ;
for (my_i = my_first_i; my_i < my_last_i; my_i++) {
   my_x = Compute_next_value( . . .);
   my_sum += my_x;
}
```

Cada core usa suas variáveis locais e executa tudo independente dos outros

Exemplo (cont.)

computação
Universidade Federal de Pelotas

- Após cada core completar sua tarefa o resultado da sua parte fica na variável my_sum que é local.

- Exemplo: 8 cores e n = 24 o retorno da função é:

1,4,3, 9,2,8, 5,1,1, 5,2,7, 2,5,0, 4,1,8, 6,5,1, 2,3,9

Exemplo (cont.)

computação
Universidade Federal de Pelotas

- Quando todos os cores tiverem terminado seus resultados estaram em my_sum.
- Já a soma global é feita por um core mestre:

```
if (I'm the master core) {
    sum = my_x;
    for each core other than myself {
        receive value from core;
        sum += value;
    }
} else {
    send my_x to the master;
}
```

Exemplo (cont.)

computação
Universidade Federal de Pelotas

Core	0	1	2	3	4	5	6	7
my_sum	8	19	7	15	7	13	12	14

Soma global

8 + 19 + 7 + 15 + 7 + 13 + 12 + 14 = 95

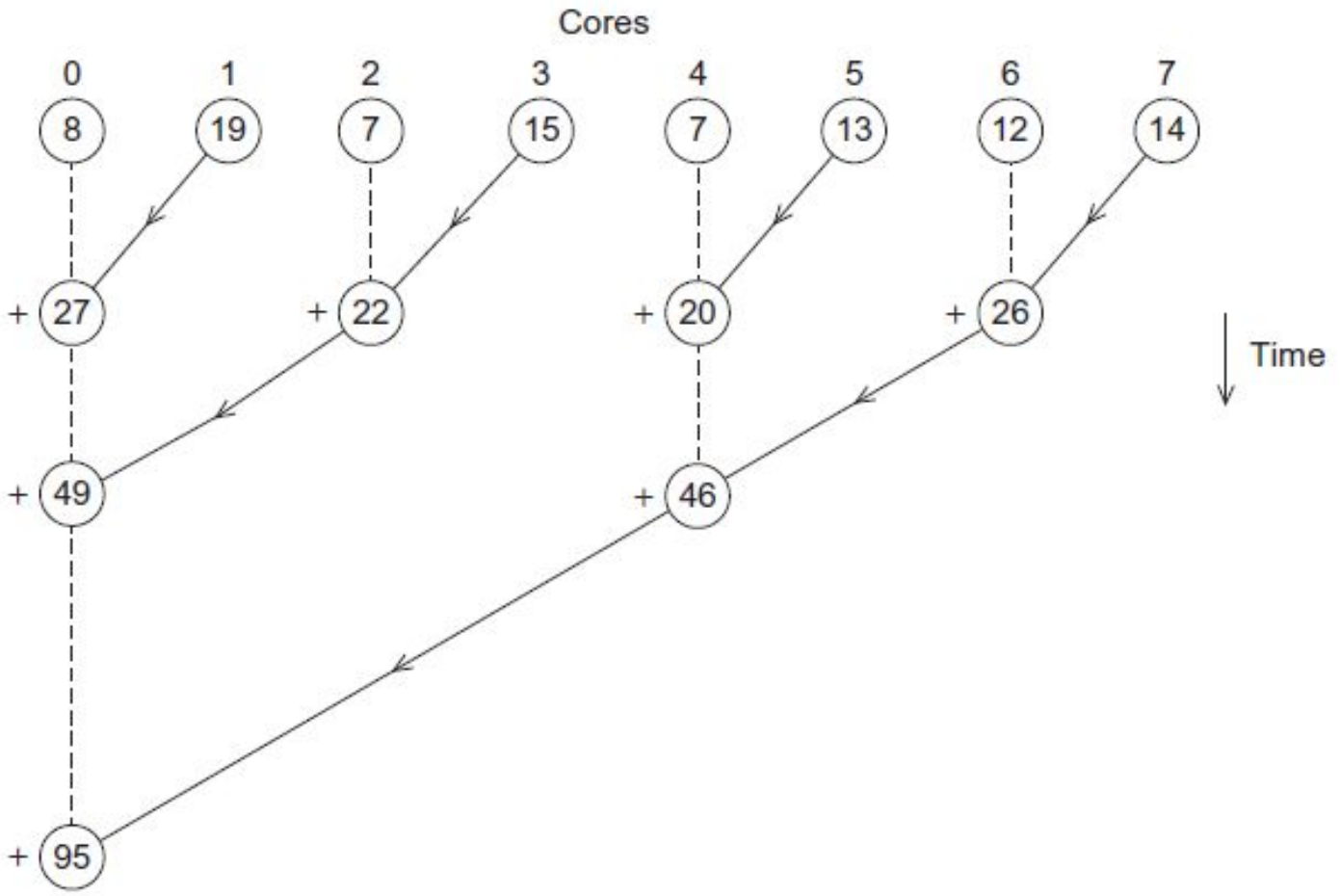
Existe uma solução melhor?

Core	0	1	2	3	4	5	6	7
my_sum	95	19	7	15	7	13	12	14

Mais paralelismo!

computação
Universidade Federal de Pelotas

- C
- P
  d
- S
  s



Análise

computação
Universidade Federal de Pelotas

- No primeiro exemplo o core mestre recebeu 7 valores e fez 7 somas.
- No segundo, o mestre recebe 3 valores e faz 3 somas.

- A melhora foi numa escala maior que 2!

Análise

computação
Universidade Federal de Pelotas

- A diferença é ainda maior com mais cores
- Com 1000 cores:
  - No primeiro exemplo o mestre receberia 999 valores e faria 999 somas.
  - No segundo ele recebe 10 valores e faz 10 somas.

- Um aumento de quase 100 vezes!

Como escrever programas paralelos?

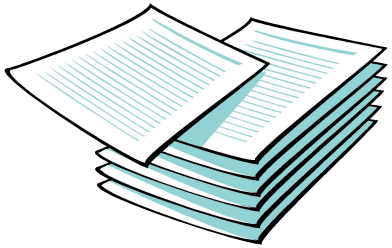
- Divisão de tarefas
  - Cada core resolve uma parte de um problema maior, não necessariamente cada core irá resolver o mesmo problema.

- Divisão dos dados
  - Dividi os dados entre os cores
  - Geralmente cada core executa o mesmo algoritmo

Exemplo: Correção de provas

computação
Universidade Federal de Pelotas

- O professor P aplicou 300 provas com 15 questões



Monitores

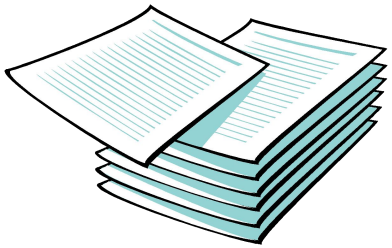
computação
Universidade Federal de Pelotas

- O professor pediu para os monitores corrigirem as provas

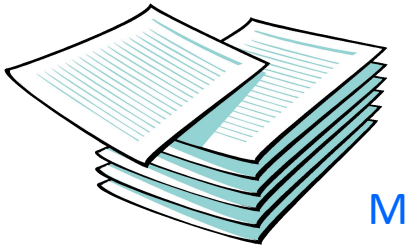
Paralelismo por divisão de dados

computação
Universidade Federal de Pelotas

MO#1

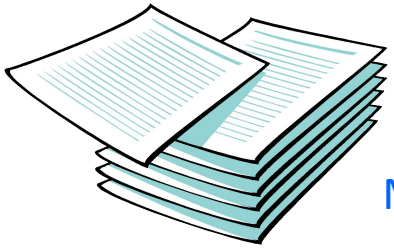


100 provas



MO#3

100 provas



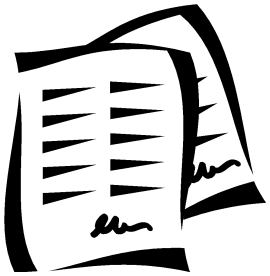
MO#2

100 provas

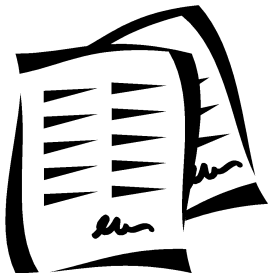
Paralelismo por divisão de tarefa

computação
Universidade Federal de Pelotas

MO#1



Questões 1 - 5



MO#3

Questões 11 - 15

MO#2

Questões 6 - 10

Coordenação

computação
Universidade Federal de Pelotas

- Geralmente precisamos coordenador o trabalho entre os cores
- Comunicação – um ou mais cores envia o seu resultado para outro core
- Balanço de carga – dividir igualmente o trabalho entre os core
- Sincronização – nem sempre os cores terminam a tarefa ao mesmo tempo é preciso sincronizar

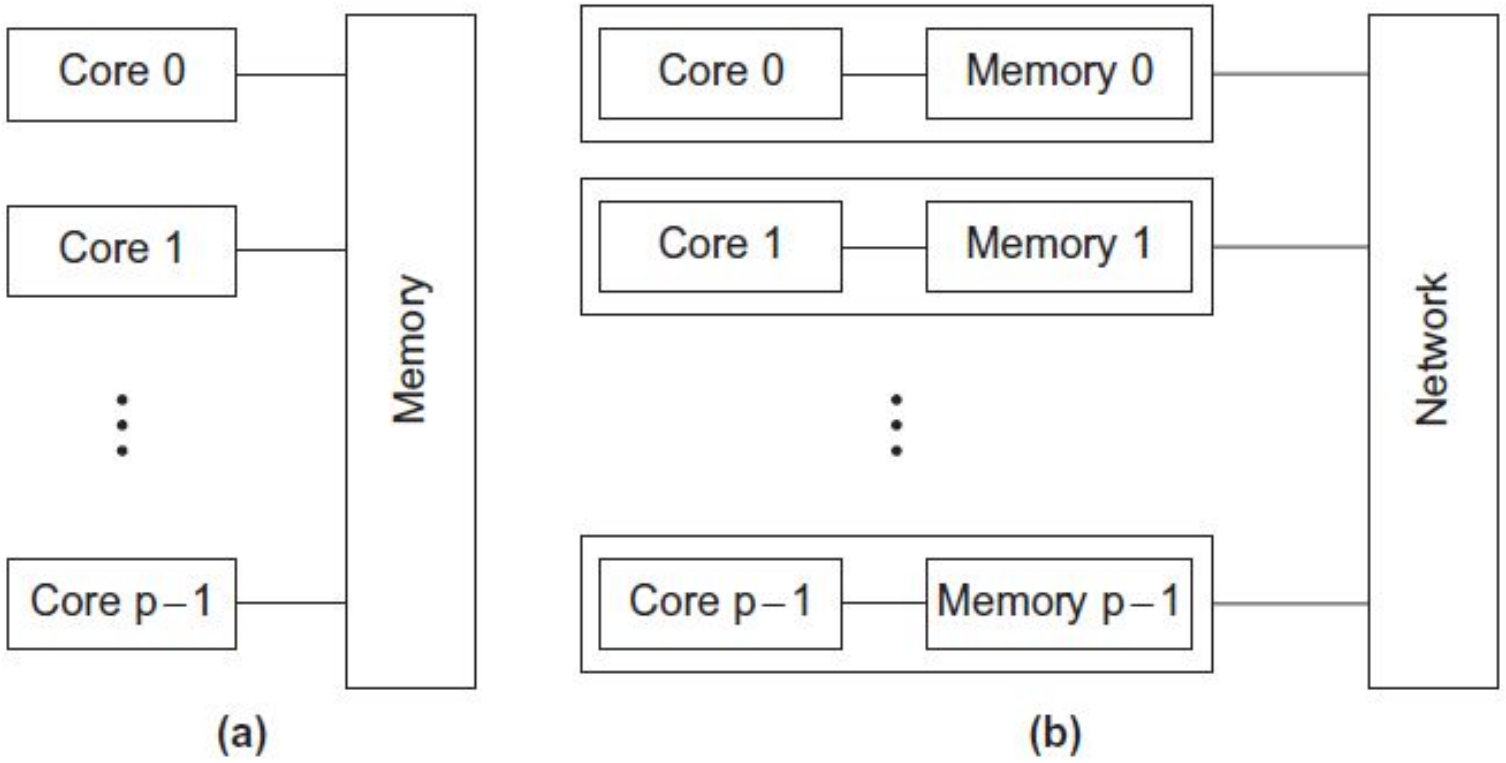
Tipos de sistemas paralelos

computação
Universidade Federal de Pelotas

- Memória compartilhada
  - Os cores usam a mesma memória
  - A coordenação é feita com os endereços de memória
- Memória distribuída
  - Cada core tem sua memória
  - Eles precisam conversar atráves de mensagens entre eles, geralmente através de uma rede

Tipos de sistemas paralelos

computação
Universidade Federal de Pelotas



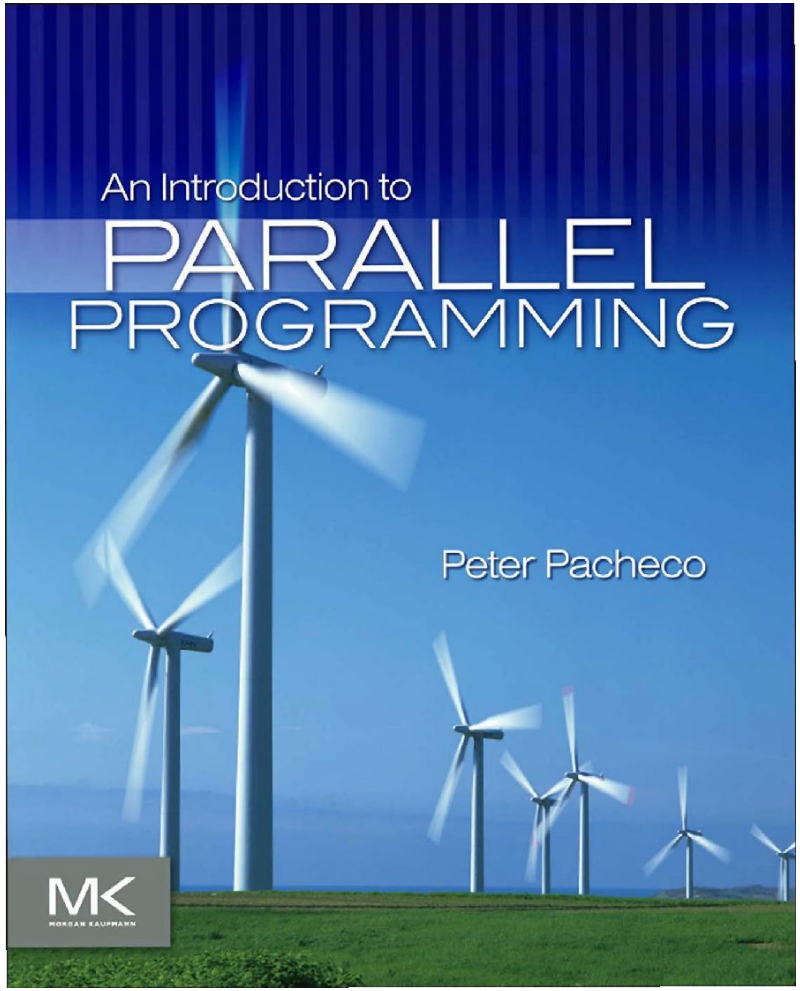
Terminologia

computação
Universidade Federal de Pelotas

- **Computação concorrente** – um programa onde múltiplas tarefas podem estar em progresso ao mesmo tempo
- **Computação paralela** – um programa onde múltiplas tarefas estão cooperando para resolver um problema
- **Computação distribuída** – um programa pode ter que cooperar com outros programas para resolver um problema

Livro texto para a introdução ao processamento paralelo

computação
Universidade Federal de Pelotas



Ler capitulo 1