

SuperComputação

Aula 01 – Introdução ao curso

2021 – Engenharia

André Filipe M. Batista <andrefmb@insper.edu.br>

Hoje

- Burocracias
- Entendendo desempenho na prática



Burocracias e Avaliação

Burocracias

- Atendimentos:
 - Sexta-feira: 09:30 – 11:00
 - Sala 23 / 6º. Andar ou via Teams
- Chat do Teams não conta como meio de comunicação.
- Comunicação principal: Email
- Datas de entrega são firmes.
- Trabalhos entregues em repositórios no Github

Gabaritos e respostas

O curso não tem gabaritos e respostas dos exercícios. Isto tem duas razões pedagógicas:

1. Copiar e colar atrapalha memorização e cria ilusão de aprendizado.
2. Curso foca em algoritmos e em sua implementação eficiente.

Gabaritos e respostas

Para cada aluno acompanhar seu progresso será oferecido:

1. Arquivos com entrada e saída esperada para todo exercício. Alguns virão com testes automatizados;
2. Algoritmos em pseudo-código.

Isso é tudo que um engenheiro da computação precisa para checar se sua solução está correta.

Objetivos de aprendizagem

1. Criar implementações eficientes para problemas computacionalmente difíceis;
2. Planejar e projetar sistemas de computação de alto desempenho, escolhendo as tecnologias mais adequadas para cada tipo de aplicação;
3. Utilizar recursos de computação multi-core para melhorar o desempenho de programas sequenciais;
4. Implementar algoritmos ingenuamente paralelizáveis em GPU;
5. Analisar resultados de desempenho levando em conta complexidade computacional e tecnologias usadas na implementação.

Objetivos de aprendizagem

1. Criar **implementações eficientes** para problemas computacionalmente difíceis;
2. Planejar e projetar sistemas de computação de alto desempenho, **escolhendo as tecnologias mais adequadas** para cada tipo de aplicação;
3. Utilizar recursos de **computação multi-core** para melhorar o desempenho de programas sequenciais;
4. Implementar algoritmos ingenuamente paralelizáveis em **GPU**;
5. Analisar **resultados de desempenho** levando em conta **complexidade computacional e tecnologias usadas** na implementação.

Avaliação

Média Final:

- Projeto = 55%
- Provas = 45%

Condições:

1. Média provas $\geq 4,5$
2. PI e PF ≥ 4
- 9 3. Projeto ≥ 5

Avaliação (DELTA provas)

Se $(PI < 4 \text{ E } PF \geq 5)$ OU $(PI \geq 5 \text{ E } PF < 4)$:

1. Aluno faz uma nova prova PD no dia da SUB relativa a avaliação em que tirou nota menor que 4.
2. Critério de barreira de provas é cumprido se $PD \geq 5$.

Avaliação (DELTA provas)

Se $(PI < 4 \text{ E } PF \geq 5)$ OU $(PI \geq 5 \text{ E } PF < 4)$:

1. Aluno faz uma nova prova PD no dia da SUB relativa a avaliação em que tirou nota menor que 4.
2. Critério de barreira de provas é cumprido se $PD \geq 5$.

Espírito da regra: se foi mal em uma prova e se recuperou na outra, merece segunda chance.

Avaliação (Datas)

- Datas são firmes. Todo atraso significa desconto de 1,0;

Nenhum dos descontos causa reprovação!

- Esses descontos nunca deixam uma nota de projeto menor que o mínimo para a aprovação (desde que seja aprovado em provas).

Gabaritos e respostas

Para cada aluno acompanhar seu progresso será oferecido:

1. Arquivos com entrada e saída esperada para todo exercício. Alguns virão com testes automatizados;
2. Algoritmos em pseudo-código.

Isso é tudo que um engenheiro da computação precisa para checar se sua solução está correta.

Gabaritos e respostas

O curso não tem gabaritos e respostas dos exercícios. Isto tem duas razões pedagógicas:

1. Copiar e colar atrapalha memorização e cria ilusão de aprendizado.
2. Curso foca em algoritmos e em sua implementação eficiente.

Ferramentas

- GCC 8.0 (ou superior) -- C++11
- Linux (Ubuntu 18.04 ou superior)
- Monstrão (containers/VMs)
 - ambiente de testes padrão



Desempenho na prática

Discussão

"Algoritmos complexos são aplicados em diversas situações para orientar decisões de negócios e para otimizar a alocação / distribuição de recursos.

Um determinado algoritmo é atualmente considerado lento demais. E agora?

Solução de alto desempenho

1. Algoritmos eficientes

2. Implementação eficiente

- Cache, paralelismo de instrução
- Linguagem de programação adequada

3. Paralelismo

Solução de alto desempenho

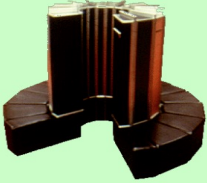


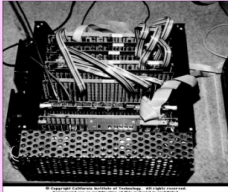

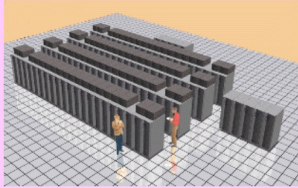

1. Algoritmos eficientes

2. Implementação eficiente

- Cache, paralelismo de instrução
- Linguagem de programação adequada

3. Paralelismo

Soluções encontradas (Paralelismo)

 Cray 1 (1976)	 Cray 2 (1985)	 Cray C-90 (1991)	Vector Computers SMP computers
 Cosmic cube (1983)	 Paragon (1993)		Massively Parallel Processors (MPP)
	 ASCI Red (1997)		
 Clusters (late 80's)			Cluster Computers Linux PC Clusters (~1995)



Como programar para estes computadores/arquiteturas?

Revolução no acesso a recursos



OPENSIFT

Super Computação sob demanda!

Revolução no acesso a recursos



OPENSIFT


Melhor aproveitamento dos recursos.



Atividade prática

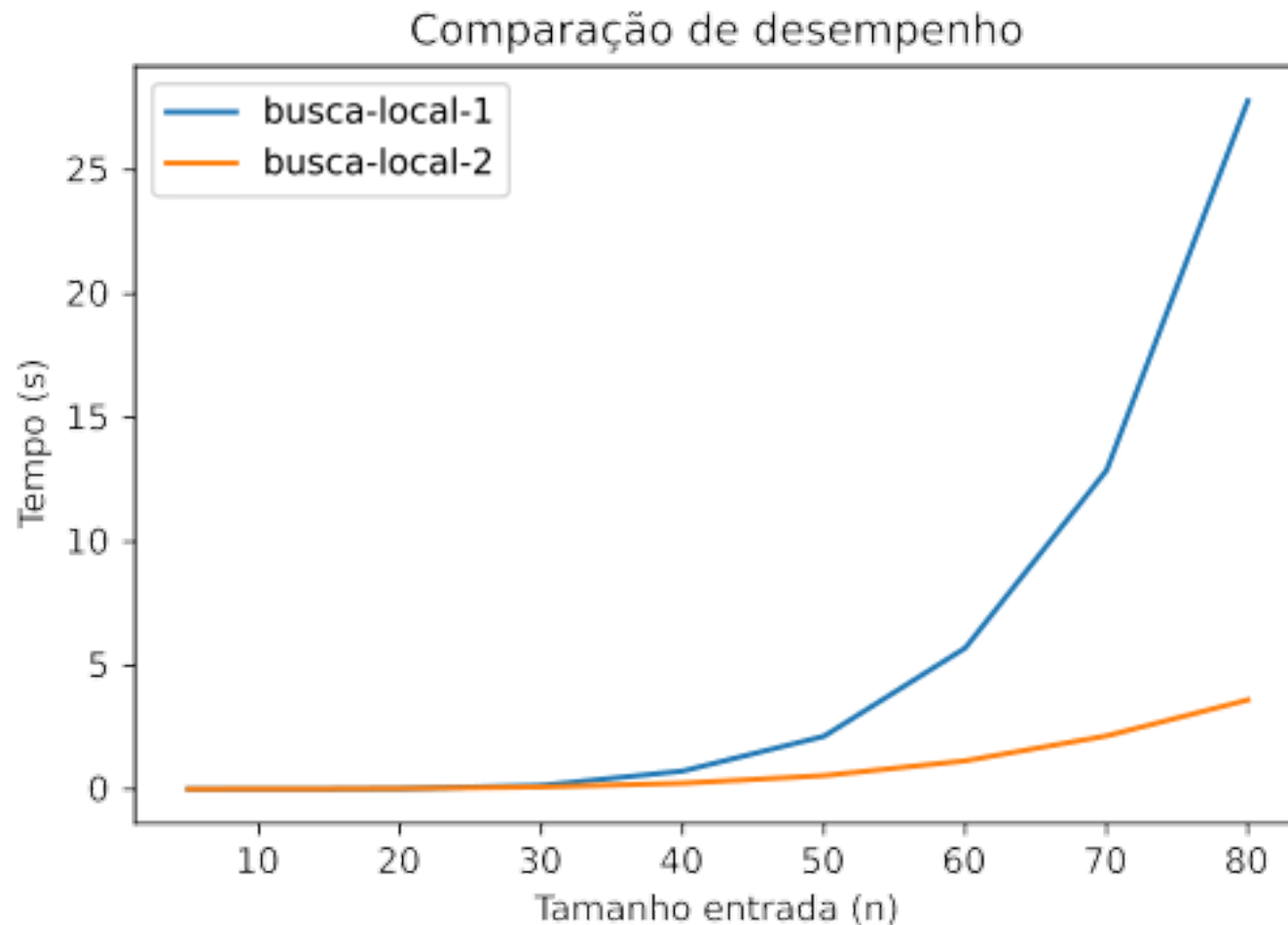
Comparando soluções para um problema

1. Medir tempo de execução de programas usando Python
2. Ordenar implementações de acordo com sua eficiência
3. Discutir custo benefício de diferentes métodos de resolução de um problema.



Discussão 1: o quanto um bom algoritmo faz diferença?

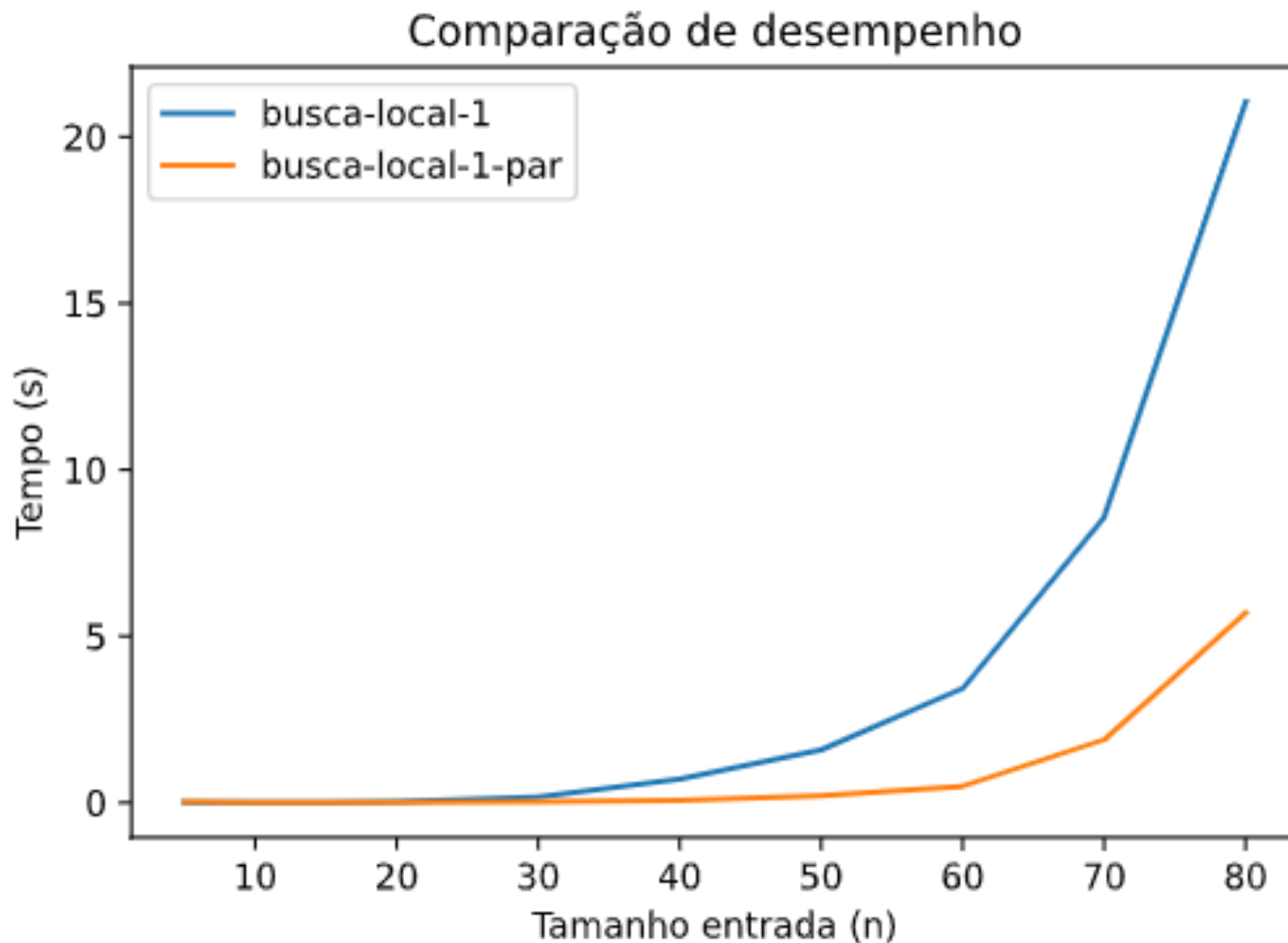
Discussão 1 - algoritmo importa!



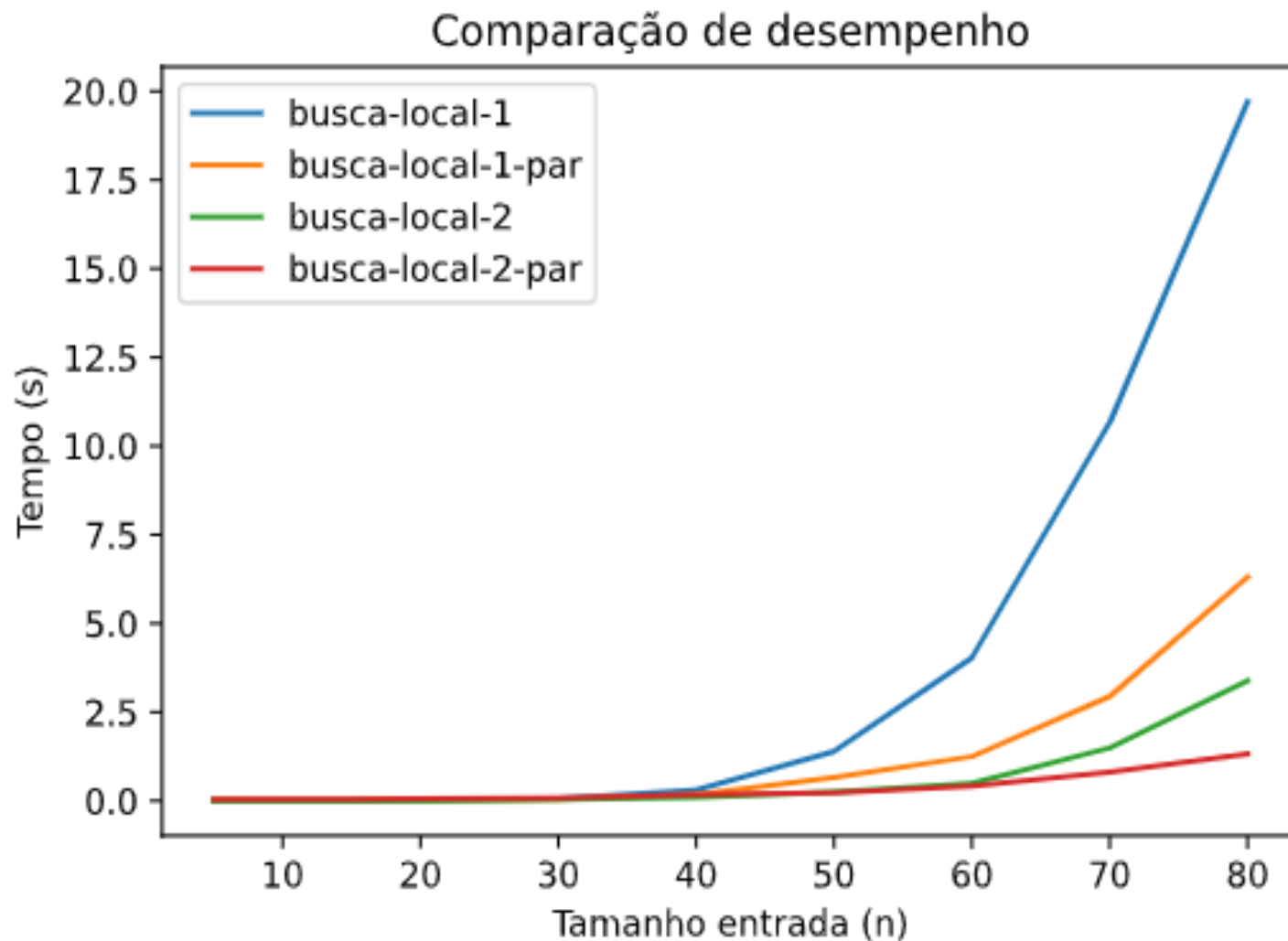


Discussão 2: o quanto paralelismo faz diferença?

Discussão 2 - paralelismo importa!



Discussão 2 - mas não sozinho....

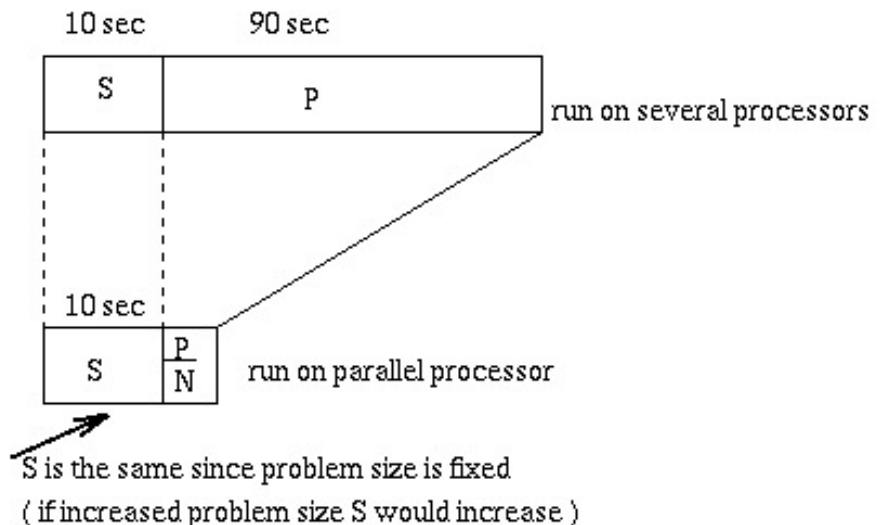


Lei de Amdahl

- Numa aplicação existe sempre uma parte que não pode ser paralelizada
- Seja **S** a parte do trabalho sequencial, **1-S** é a parte susceptível de ser paralelizada
- Mesmo que a parte paralela seja perfeitamente escalável, o aumento do desempenho (speedup) está limitado pela parte sequencial

n = número de processadores

$$\text{Speedup} = \frac{1}{S + \frac{(1 - S)}{n}}$$



Lei de Amdahl

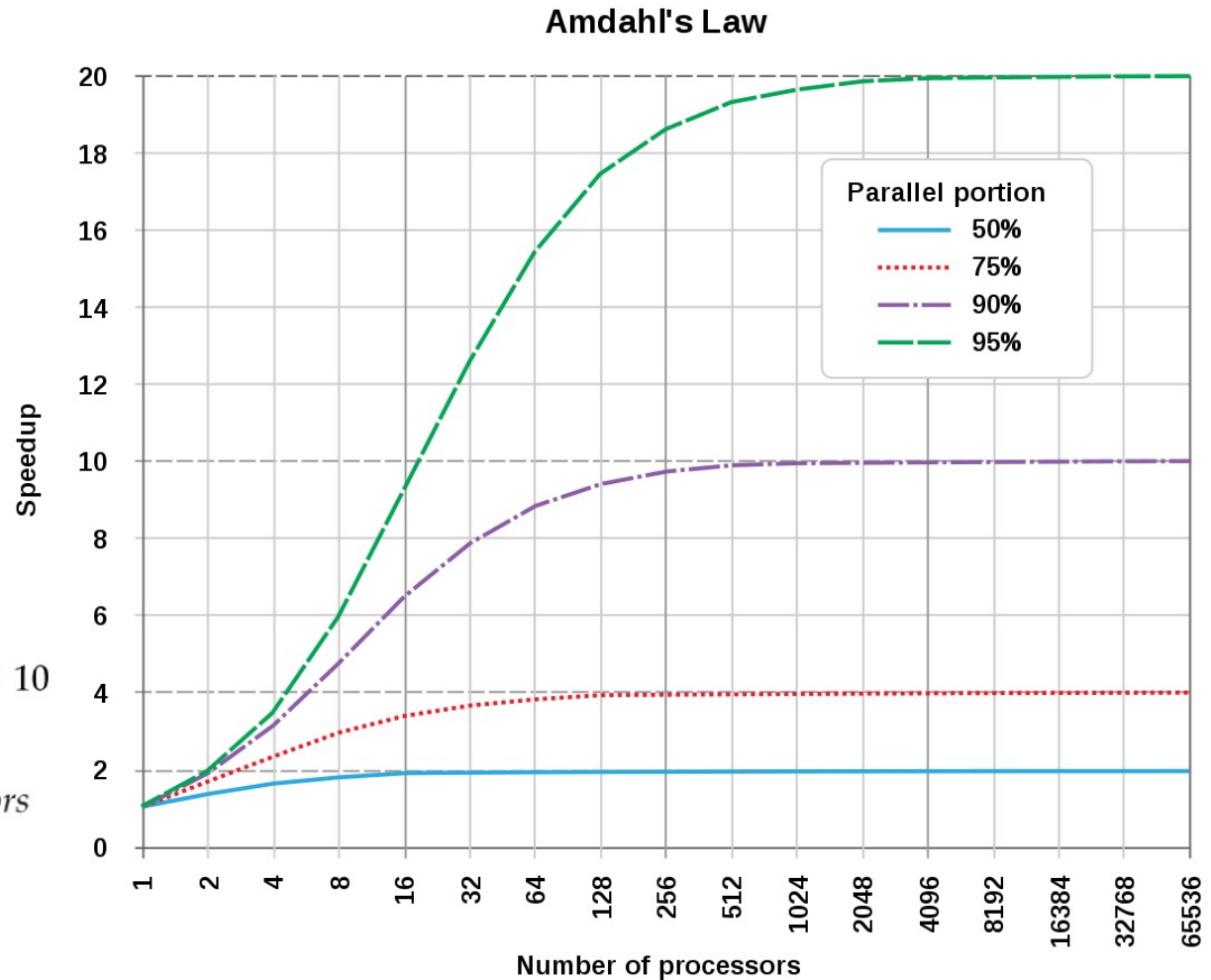
Se 10% das operações de um código precisam ser feitas sequencialmente, então o speedup não pode ser maior do que 10, independente do número de processadores

$$S = \frac{1}{0.1 + \frac{0.9}{10}} \cong 5.3$$

$p = 10$ processors

$$S = \frac{1}{0.1 + \frac{0.9}{\infty}} = 10$$

$p = \infty$ processors



Insper

www.insper.edu.br