

# SuperComputação

Aula 1 – Introdução ao curso

2020 – Engenharia

Luciano Soares <[lpsoares@insper.edu.br](mailto:lpsoares@insper.edu.br)>

Igor Montagner <[igorsm1@insper.edu.br](mailto:igorsm1@insper.edu.br)>

# Hoje

- Burocracias
- Resumo do curso
- Atividade prática de C++

# Burocracias e Avaliação

# Questão presencial/remoto

- Iremos contar presença.
- Aulas presenciais a partir de 08/09.
- Divisão de turmas tentará agradar todos
- Turma 100% remota terá aula exclusivamente pelo Teams
- Turma 100% presencial terá aulas no Insper
- Pode eventualmente trocar, mas precisa ser conversado.

# Questão presencial/remoto

## **T1 (remota):**

- SuperComputação TER 09:45 SEX 07:30
- Jogos Digitais QUA 13:30 SEX 13:30

## **T2 (presencial):**

- SuperComputação QUA 13:30 SEX 13:30
  - Jogos Digitais TER 09:45 SEX 07:30



# Atendimentos

- Atendimentos:
  - SEX 9:30 - 11:00
  - SEX 15:30 - 17:00
- Emails SEG/TER/QUA 16:00 - 17:00
  - Resposta em 2 horários. Não respondi? Me lembre.
  - Responsabilidade compartilhada.
  - E-mail é para discussão de algoritmos, não código.
- Chat do Teams não conta como meio de comunicação.
- Posso pedir no email para resolver no atendimento.

# 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  $\geq 4$
3. Projeto  $\geq 5$

# 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  $PD \geq 5$ .

# Avaliação (Projeto)

1. Rubrica D: Implementação correta de todas as partes
2. Rubrica C: Relatório feito de acordo com os critérios de aula
3. Rubrica B: Todas as implementações são minimamente eficientes.
4. +(3,0): Competição de desempenho para cada uma das três partes.

# Avaliação (Projeto - Detalhes)

- Os itens que compõe a rubrica C são obrigatórios;
- Correções parcialmente baseadas em testes automatizados;
- Competições serão feitas perto da PI e PF com configurações de hardware padrão

# Avaliação (Projeto - atrasos e descontos)

- Datas são firmes. Todo atraso significa desconto de 1,0;
- Qualidade de código é importante. Uma lista de requisitos está na página de projeto

**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).

# Ferramentas

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



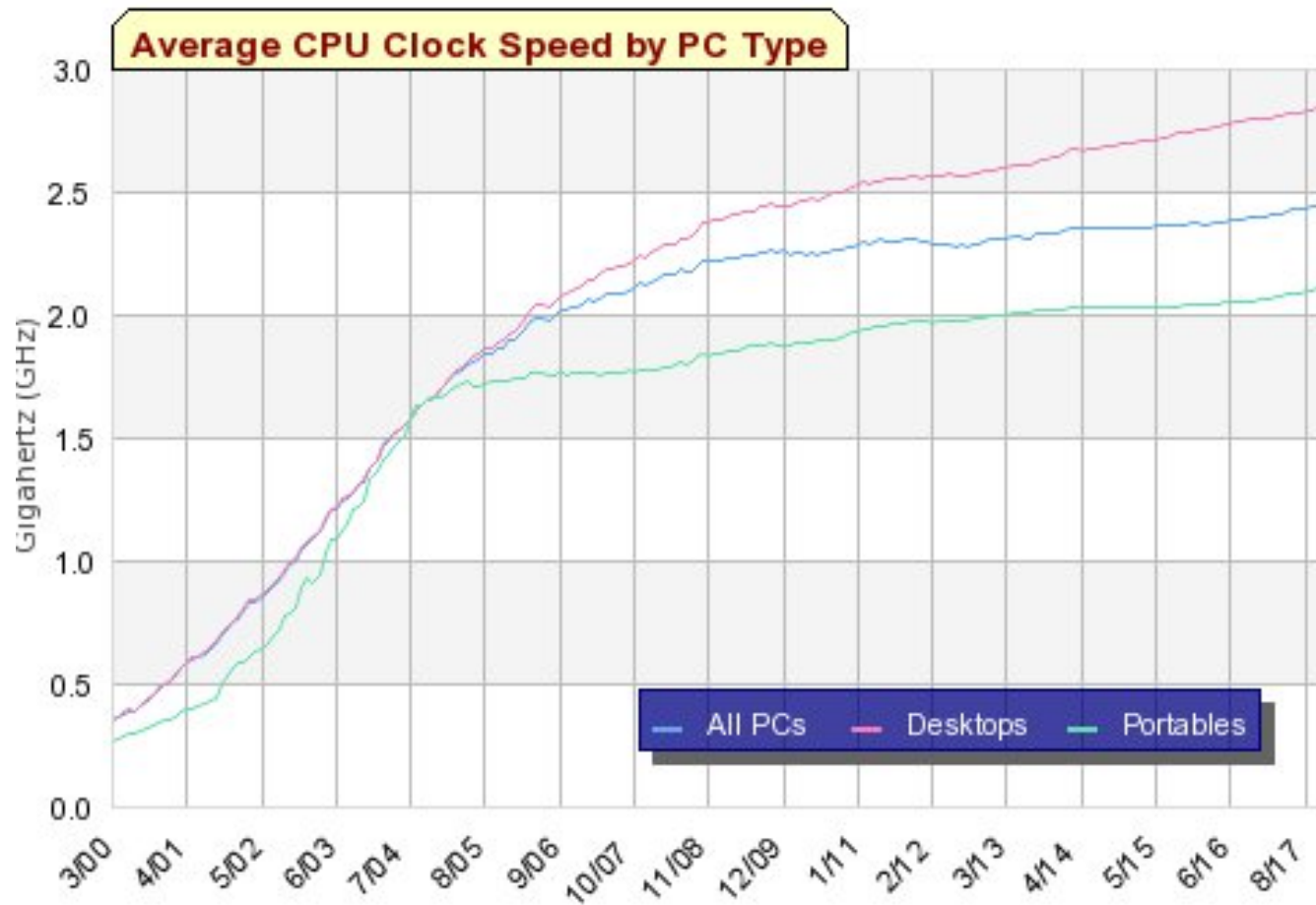
# Resumo do curso

# Objetivo de SuperComputação

Aumentar velocidade de processamento! (Hardware)

- Mais clock
- Memória mais rápida
- Mais núcleos
- Melhor resfriamento

# Objetivo de SuperComputação



# Objetivo de SuperComputação

Event	Latency	Scaled
1 CPU cycle	0.3 ns	1 s
Level 1 cache access	0.9 ns	3 s
Level 2 cache access	2.8 ns	9 s
Level 3 cache access	12.9 ns	43 s
Main memory access (DRAM, from CPU)	120 ns	6 min
Solid-state disk I/O (flash memory)	50–150 µs	2–6 days
Rotational disk I/O	1–10 ms	1–12 months
Internet: San Francisco to New York	40 ms	4 years
Internet: San Francisco to United Kingdom	81 ms	8 years
Internet: San Francisco to Australia	183 ms	19 years
TCP packet retransmit	1–3 s	105–317 years
OS virtualization system reboot	4 s	423 years
SCSI command time-out	30 s	3 millennia
Hardware (HW) virtualization system reboot	40 s	4 millennia
Physical system reboot	5 m	32 millennia

# Objetivo de SuperComputação

Aumentar velocidade de processamento! (Software)

# Objetivo de SuperComputação

Aumentar velocidade de processamento! (Software)

- Interpretador/JIT/compilador mais rápido
- Algoritmos melhores
- Melhorar organização dos dados

# Objetivo de SuperComputação

Notação	Nome	Característica	Exemplo
$O(1)$	constante	independe do tamanho $n$ da entrada	determinar se um número é par ou ímpar; usar uma tabela de dispersão (hash) de tamanho fixo
$O(\log n)$	logarítmica	o problema é dividido em problemas menores	busca binária
$O(n)$	linear	realiza uma operação para cada elemento de entrada	busca sequencial; soma de elementos de um vetor
$O(n \log n)$	log-linear	o problema é dividido em problemas menores e depois junta as soluções	heapsort, quicksort, merge sort
$O(n^2)$	quadrática	itens processados aos pares (geralmente loop aninhado)	bubble sort (pior caso); quick sort (pior caso); selection sort; insertion sort
$O(n^3)$	cúbica		multiplicação de matrizes $n \times n$ ; todas as triplas de $n$ elementos
$O(n^c, c > 1)$	polinomial		caixeiro viajante por programação dinâmica
$O(c^n)$	exponencial	força bruta	todos subconjuntos de $n$ elementos
$O(n!)$	fatorial	força bruta: testa todas as permutações possíveis	caixeiro viajante por força bruta

# Objetivo de SuperComputação

Aumentar velocidade de processamento! (Software)

- Interpretador/JIT/compilador mais rápido
- Algoritmos melhores
- Melhorar organização dos dados

Sistemas Hardware-software





# Objetivo de SuperComputação

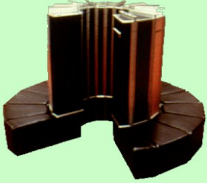


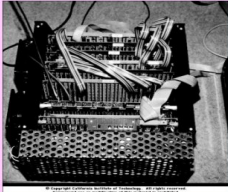

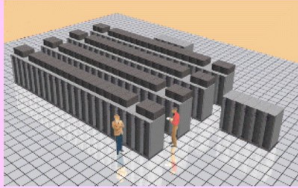

Aumentar velocidade de processamento! (Software)

- Interpretador/JIT/compilador mais rápido
- Algoritmos melhores
- Melhorar organização dos dados



Desafios de Programação

# Soluções encontradas

 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)

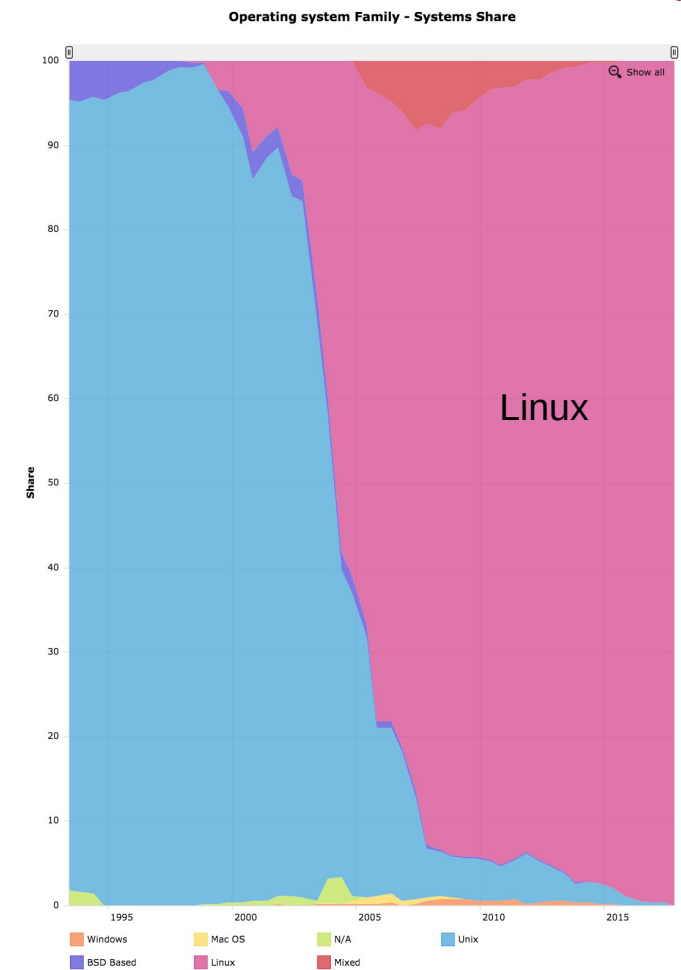
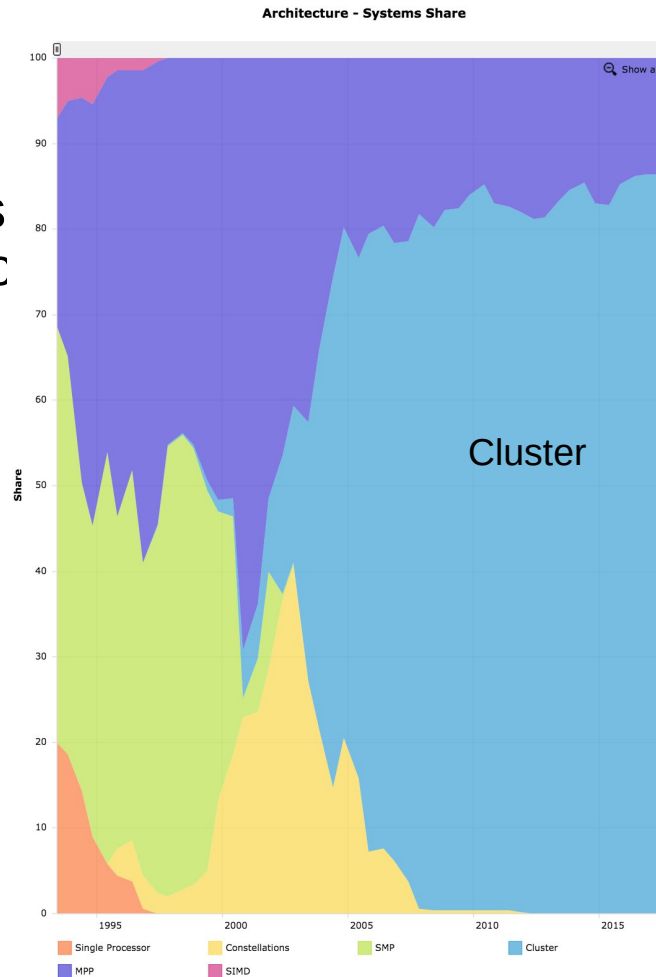
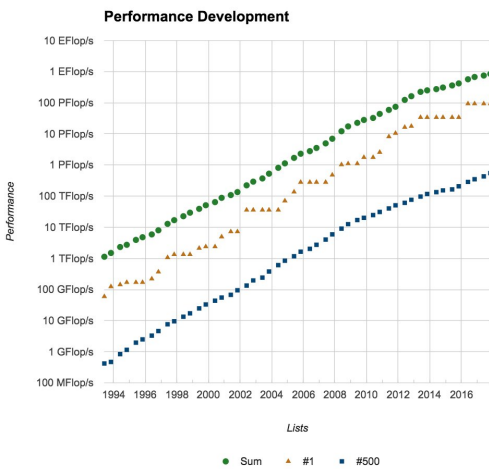


Como programar para estes computadores/arquiteturas?

# TOP 500

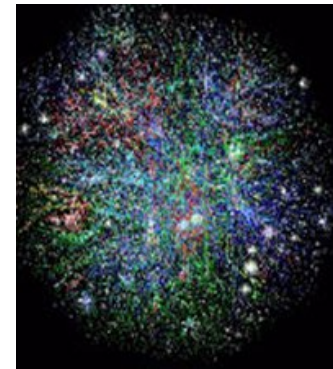
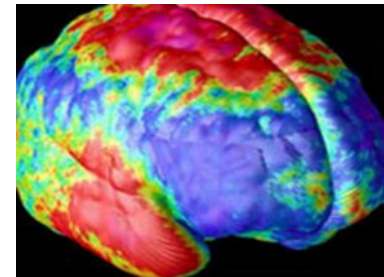
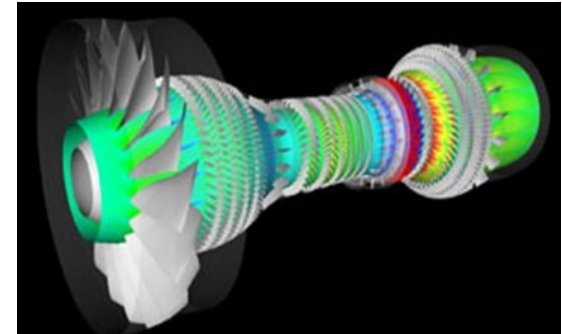
<https://www.top500.org/statistics/overtime/>

- Ranking dos 500 computadores mais poderosos do mundo.



## Aplicações de Supercomputação

- Previsão do tempo
- Cálculo de aerodinâmica e car crash
- Análise probabilística
- Modelagem de proteção contra radiação.
- Quebra de senhas por força bruta
- Simulações de testes nucleares 3D
- Simulação de Dinâmica Molecular
- Minização de consumo de combustível por rotas de entrega



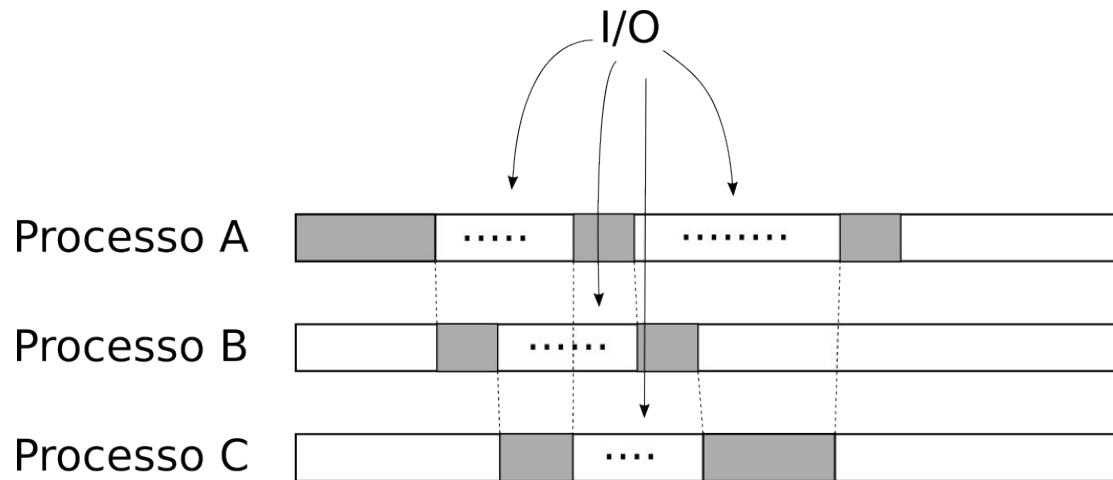
Argnome e  
<https://en.wikipedia.org/wiki/Supercomputer>

# Conteúdos

- Programação concorrente e sincronização
- Programação paralela em CPUs multi core
- GPGPU

# Programação concorrente

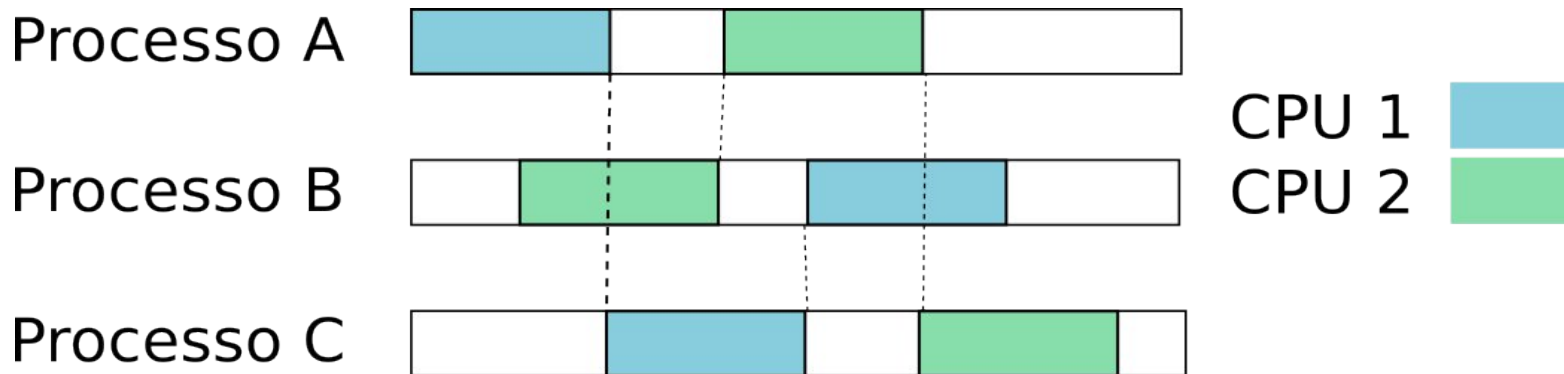
- Tarefas limitadas por entrada e saída



- Estratégia de divisão do problema em tarefas
- Primitivas para sincronizar a execução

# Programação Multi core

- Tarefas limitadas por CPU



- Divisão em partes independentes
- Modelo fork-join



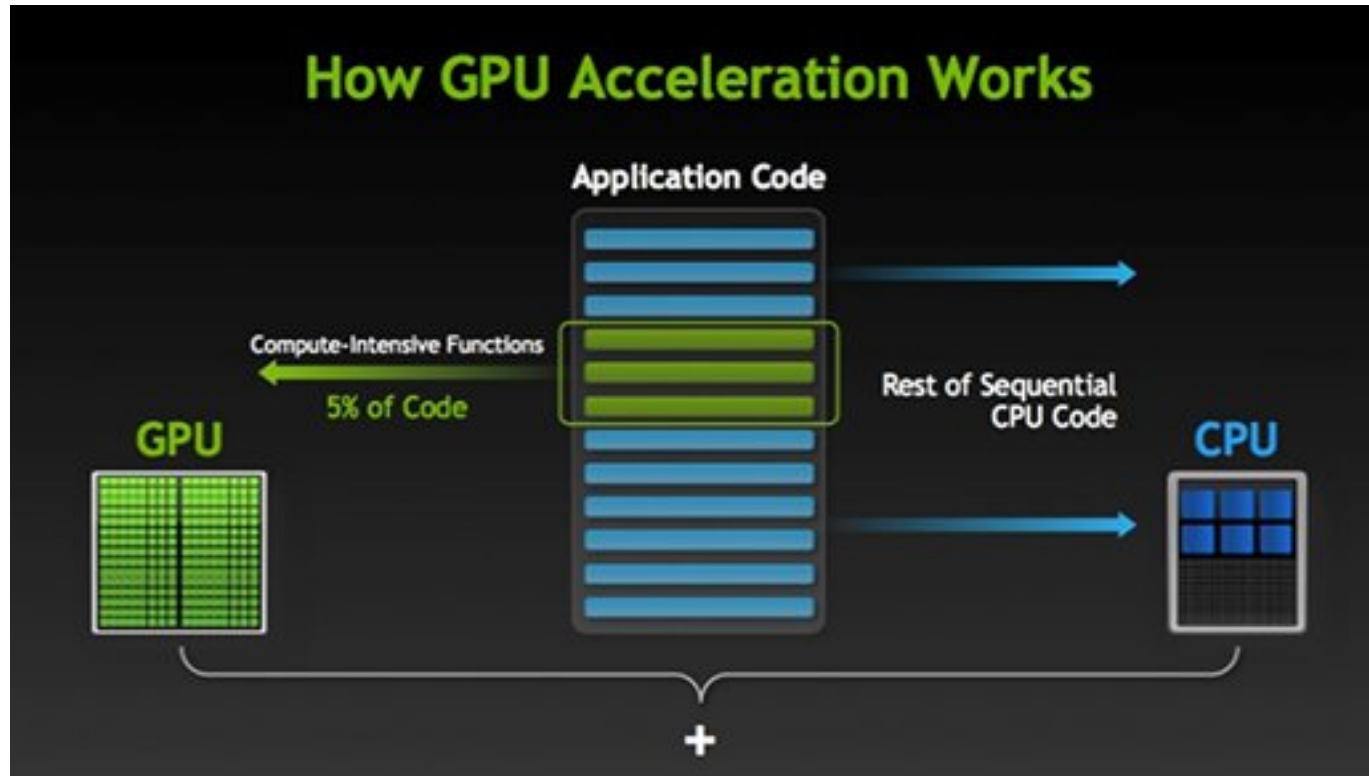
# Sistemas distribuídos



- Divisão de tarefas em clusters
- Passagem de mensagens entre processos/máquinas



# GPGPU



- Versão turbinada do modelo multi-core
- Arquitetura completamente diferente

# Revolução no acesso a recursos



**OPENSIFT**

**Super Computação sob  
demanda!**

# Visão geral do curso

Estratégias para resolução de problemas difíceis

- Complexidade Computacional
- Problemas NP-completo
- Heurísticas
- Busca local e global
- Algoritmo aleatorizados

PI

PF

- Paralelismo
- Sistemas Multi-core
- GPU
- Projeto de programas paralelos

Processamento paralelo



# Atividade prática

## Recursos de C++:

1. Implementação de algoritmos simples
2. Recursos úteis de C++

# 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.**

# Insper

[www.insper.edu.br](http://www.insper.edu.br)