

A Necessidade da Computação de Alto Desempenho para os Dias Atuais

Jeremias Moreira Gomes

jeremiasmg@gmail.com

29 de setembro de 2016

Sumário

- 1 Introdução
- 2 Computação de Alto Desempenho
- 3 Como Funciona?
- 4 Exemplos de Uso
- 5 Conclusão
- 6 Referências Bibliográficas

O que é Computação de Alto Desempenho?

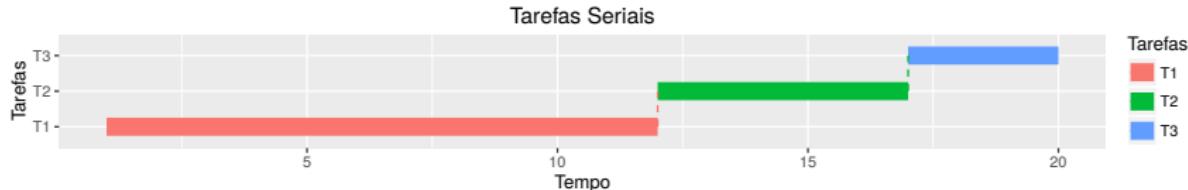
Definição

É a prática de agregar poder computacional para proporcionar um alto desempenho, a fim de resolver problemas de ciência, engenharia ou negócios [1].

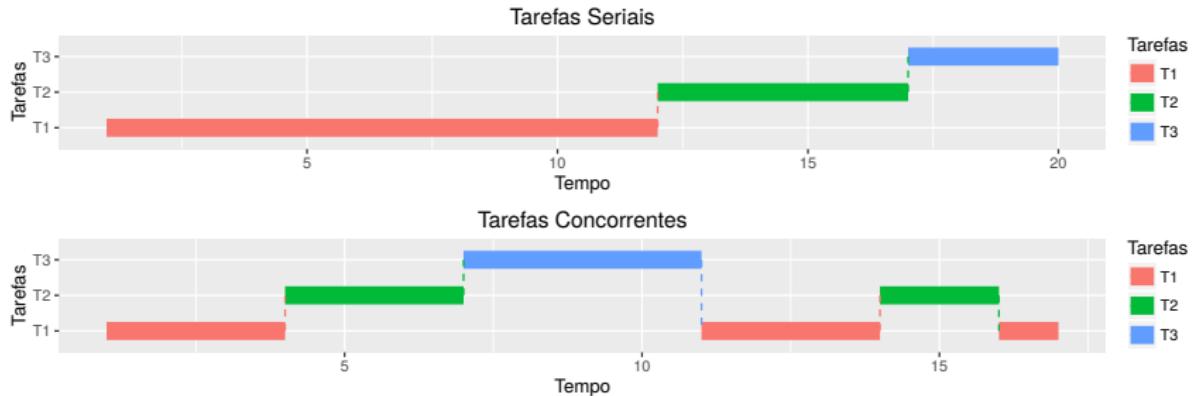
O que *NÃO* é Computação de Alto Desempenho

Concorrência vs Paralelismo

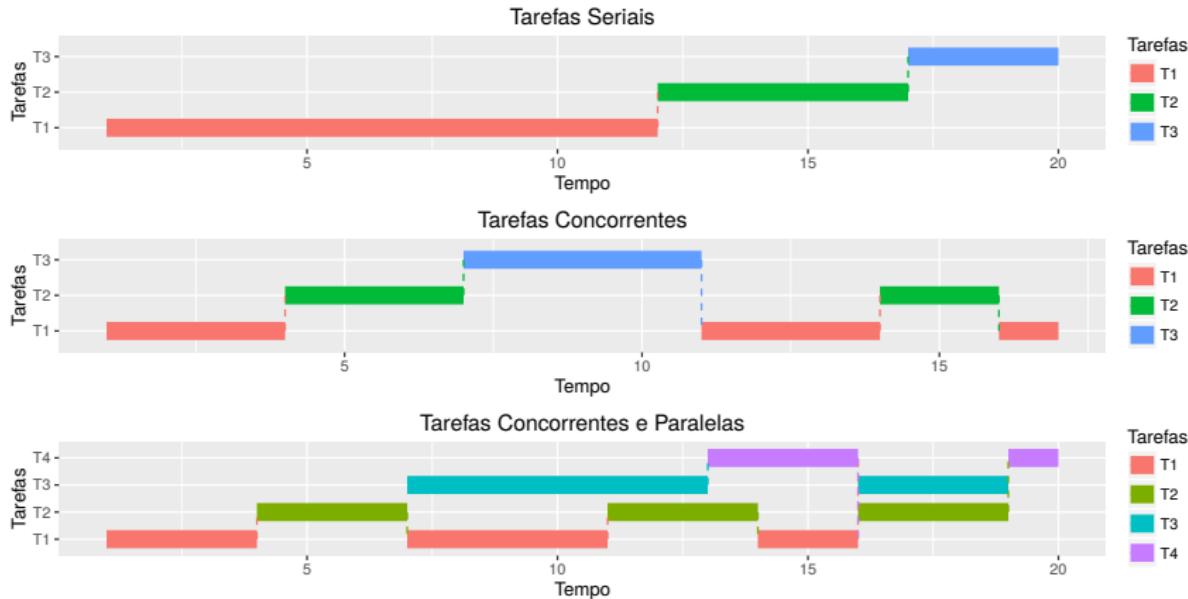
O que NÃO é Computação de Alto Desempenho



O que NÃO é Computação de Alto Desempenho



O que NÃO é Computação de Alto Desempenho



Como Executar uma Tarefa Mais Rápido?!?

Essencialmente, há três maneiras de se executar uma tarefa de maneira mais rápida

Como Executar uma Tarefa Mais Rápido?!?

Essencialmente, há três maneiras de se executar uma tarefa de maneira mais rápida

- Trabalhando mais rápido

Como Executar uma Tarefa Mais Rápido?!?

Essencialmente, há três maneiras de se executar uma tarefa de maneira mais rápida

- Trabalhando mais rápido
- Trabalhando de maneira mais inteligente

Como Executar uma Tarefa Mais Rápido?!?

Essencialmente, há três maneiras de se executar uma tarefa de maneira mais rápida

- Trabalhando mais rápido
- Trabalhando de maneira mais inteligente
- Solicitando apoio

Trabalhando Mais Rápido

Aumentando a Velocidade da CPU

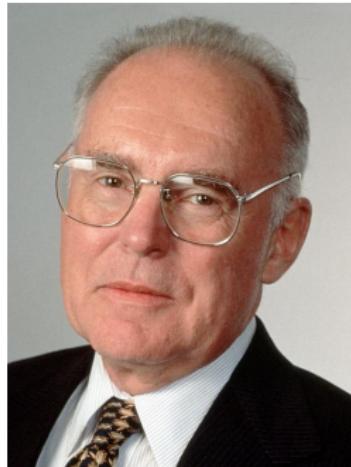
- Lei de Moore
- Funcionou muito bem até 2005
- Barreira tecnológica

Trabalhando Mais Rápido

Aumentando a Velocidade da CPU

- **Lei de Moore**
- Funcionou muito bem até 2005
- Barreira tecnológica

Lei de Moore



- Bacharel em Química pela Universidade da Califórnia (1950)
- Bacharel em Física pela Caltech (1954)
- Cofundador da Intel Corporation (1968)
- Fez uma previsão sobre o futuro do hardware conhecida como “Lei de Moore” [2]

Figura : Gordon E.
Moore

Lei de Moore

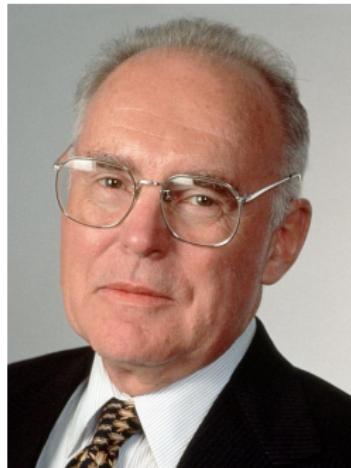
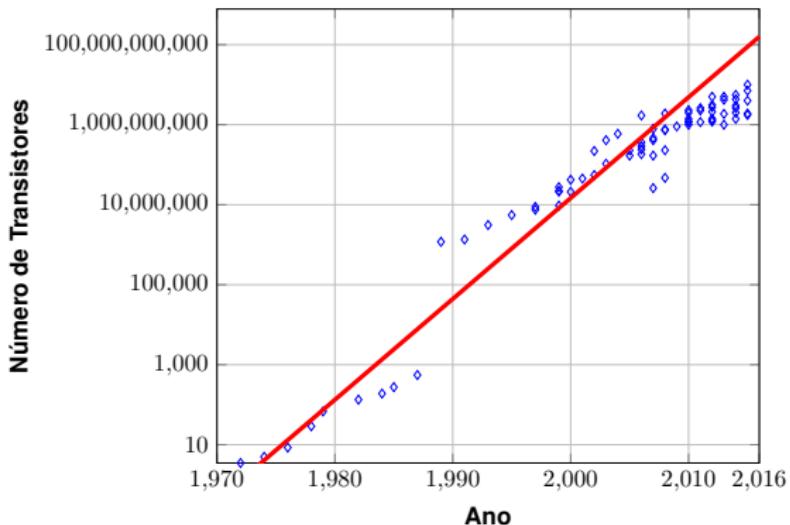


Figura : Gordon E.
Moore

- Bacharel em Química pela Universidade da Califórnia (1950)
- Bacharel em Física pela Caltech (1954)
- Cofundador da Intel Corporation (1968)
- Fez uma previsão sobre o futuro do hardware conhecida como “Lei de Moore” [2]

“A complexidade para componentes com custos mínimos tem aumentado em uma taxa de aproximadamente um fator de dois por ano.”

Evolução da Quantidade de Transistores nos Processadores



Trabalhando Mais Rápido

Aumentando a Velocidade da CPU

- Lei de Moore
- Funcionou muito bem até 2005
- Barreira tecnológica

Performance Computacional

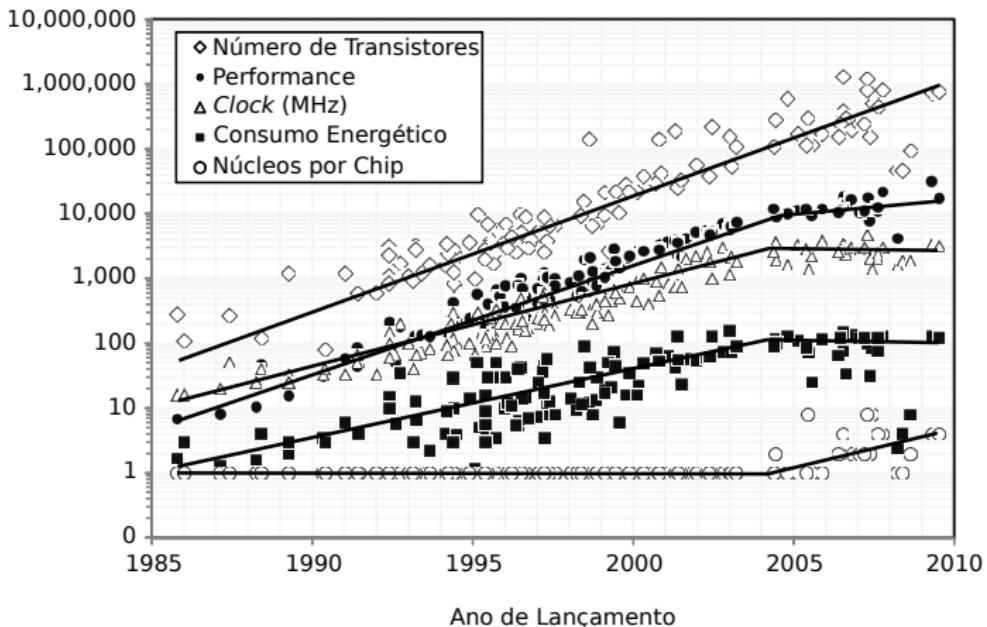


Figura : Evolução da Performance Computacional. Adaptado de [3]

Performance Computacional

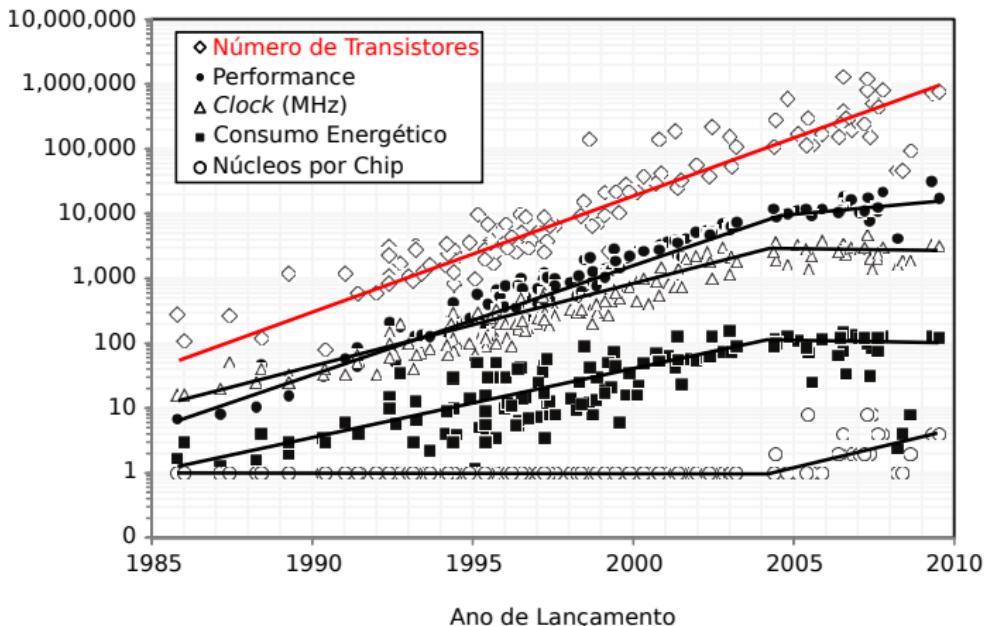


Figura : Evolução da Performance Computacional. Adaptado de [3]

Performance Computacional

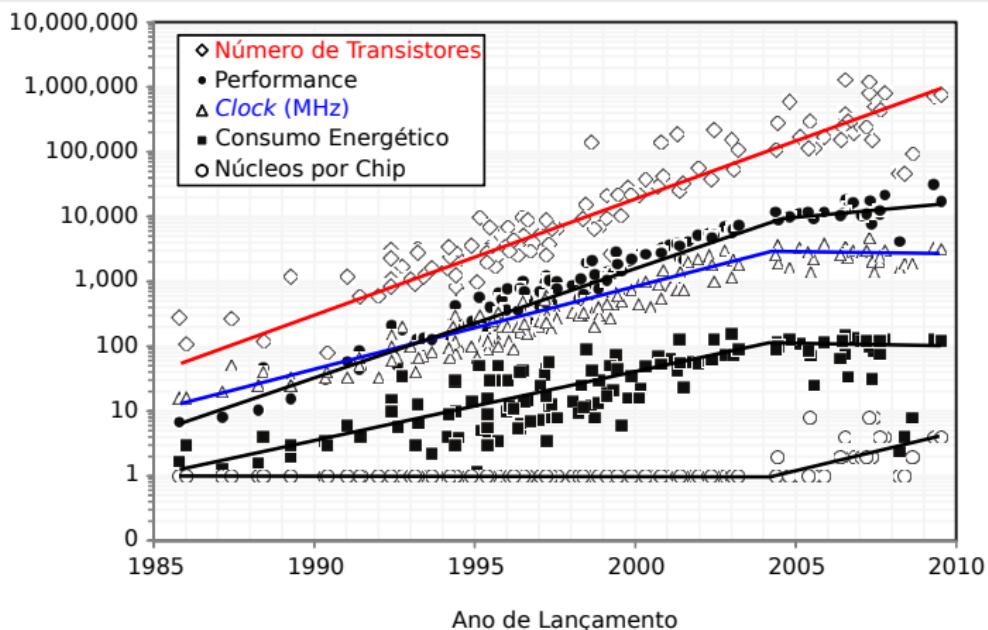


Figura : Evolução da Performance Computacional. Adaptado de [3]

Performance Computacional

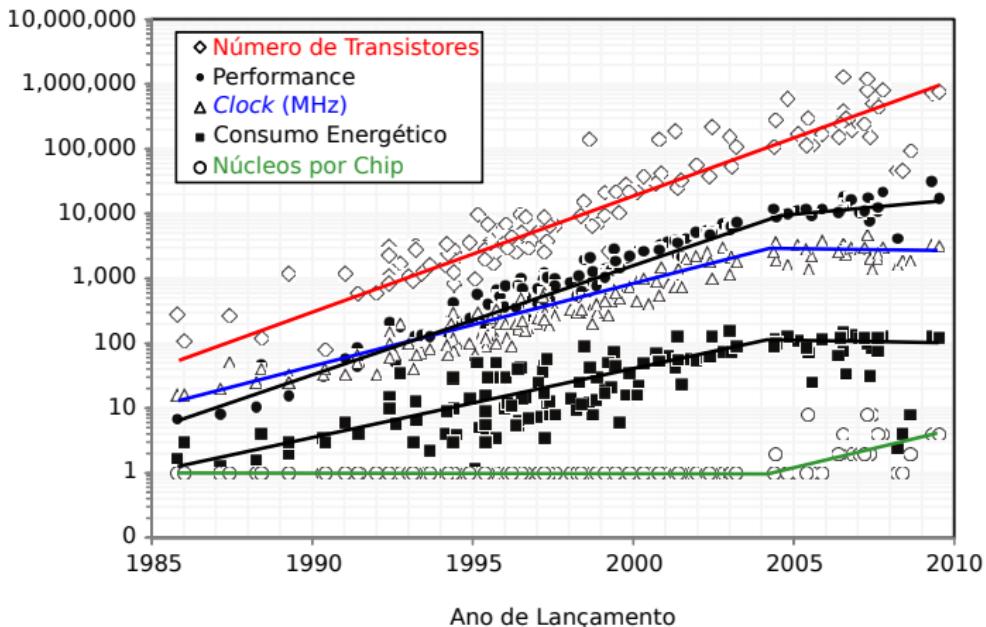


Figura : Evolução da Performance Computacional. Adaptado de [3]

Número de Núcleos por Soquete

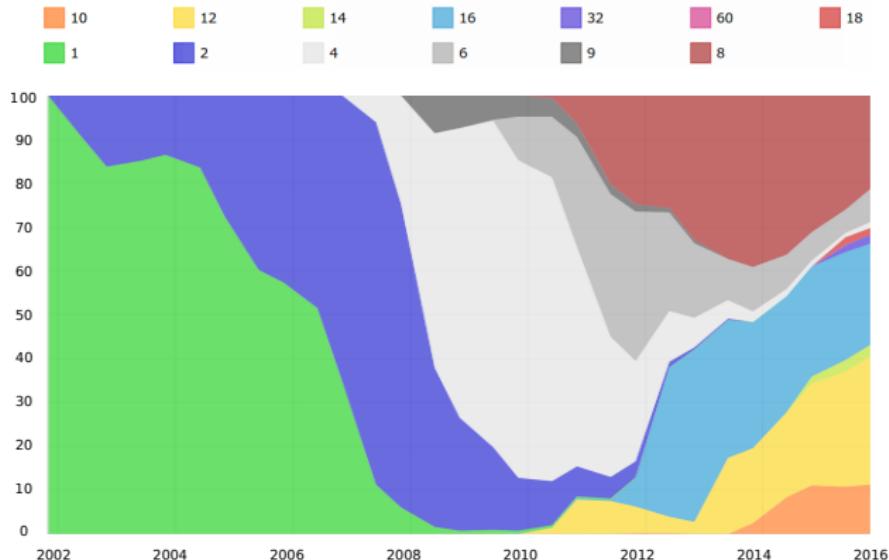


Figura : Evolução da Quantidade de Núcleos. Retirado de [4].

Trabalhando Mais Rápido

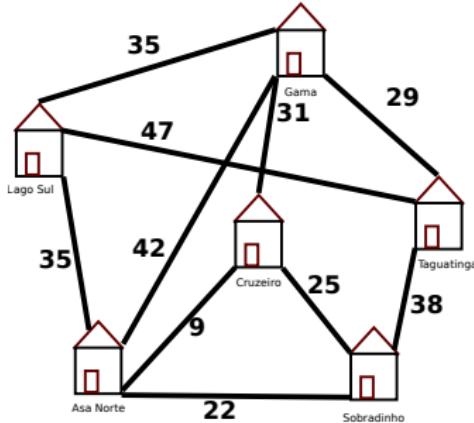
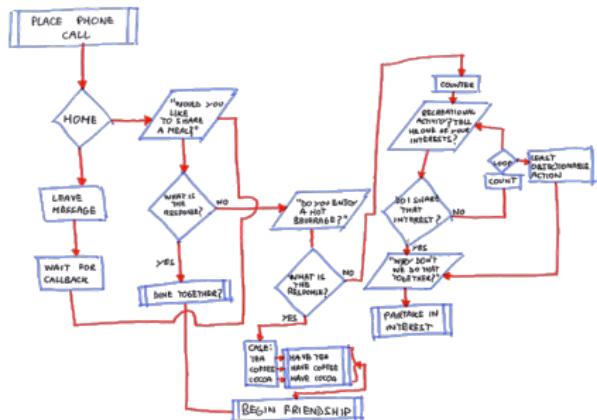
Aumentando a Velocidade da CPU

- Lei de Moore
- Funcionou muito bem até 2005
- Barreira tecnológica

Trabalhando de Maneira Mais Inteligente

Mudança de Abordagem

- Escolha apropriada do algoritmo
- Adaptações em algoritmos
- Problemas sem solução polinomial determinística (NP)



Solicitando Apoio

Buscamos de alguma forma executar as tarefas em paralelo

Computação de Alto Desempenho

- Paralelismo encoberto
 - Único processador executa diferentes instruções simultaneamente
- Paralelismo aberto
 - Programador manipula fluxos diferentes de instruções



Serial

Solicitando Apoio

Buscamos de alguma forma executar as tarefas em paralelo

Computação de Alto Desempenho

- Paralelismo encoberto
 - Único processador executa diferentes instruções simultaneamente
- Paralelismo aberto
 - Programador manipula fluxos diferentes de instruções



Serial



Encoberto

Solicitando Apoio

Buscamos de alguma forma executar as tarefas em paralelo

Computação de Alto Desempenho

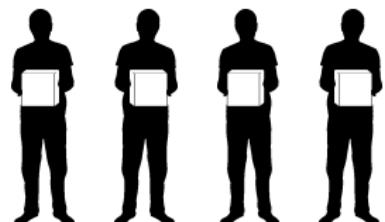
- Paralelismo encoberto
 - Único processador executa diferentes instruções simultaneamente
- Paralelismo aberto
 - Programador manipula fluxos diferentes de instruções



Serial



Encoberto



Aberto

Taxonomia de Flynn

Modelo para classificação de arquiteturas mais aceito pela comunidade científica [5].

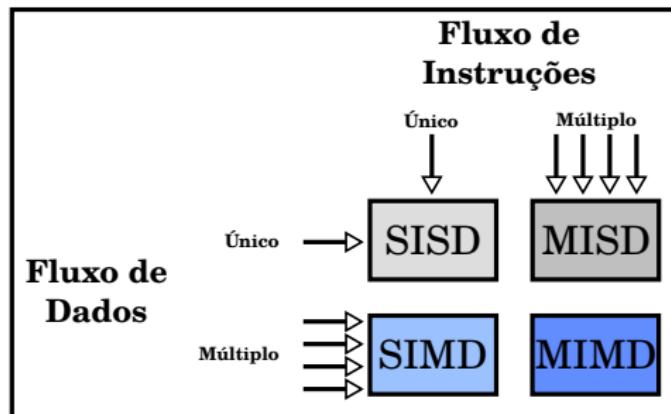


Figura : Taxonomia de Flynn

Taxonomia de Flynn

Modelo para classificação de arquiteturas mais aceito pela comunidade científica [5].

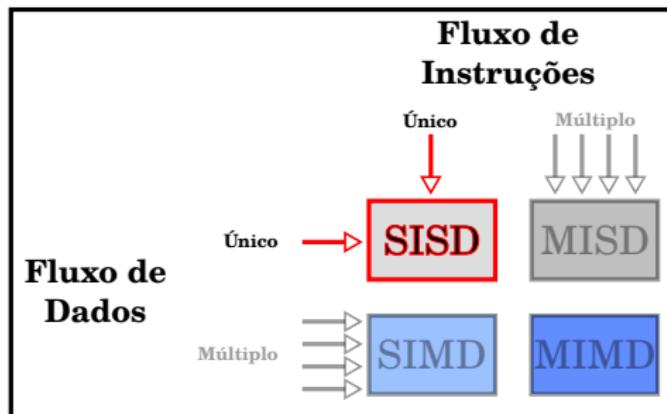


Figura : Single Instruction, Single Data

Taxonomia de Flynn

Modelo para classificação de arquiteturas mais aceito pela comunidade científica [5].

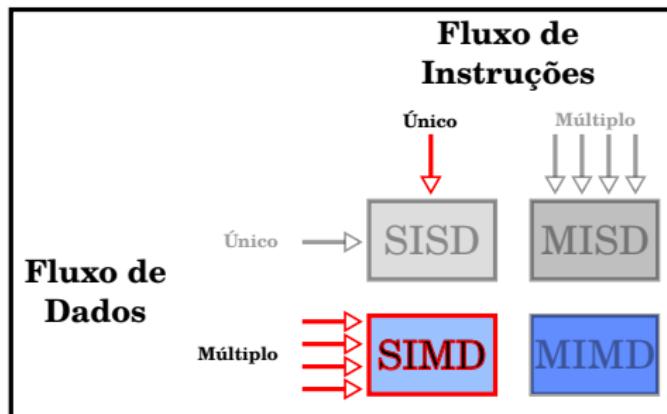


Figura : *Single Instruction, Multiple Data*

Taxonomia de Flynn

Modelo para classificação de arquiteturas mais aceito pela comunidade científica [5].

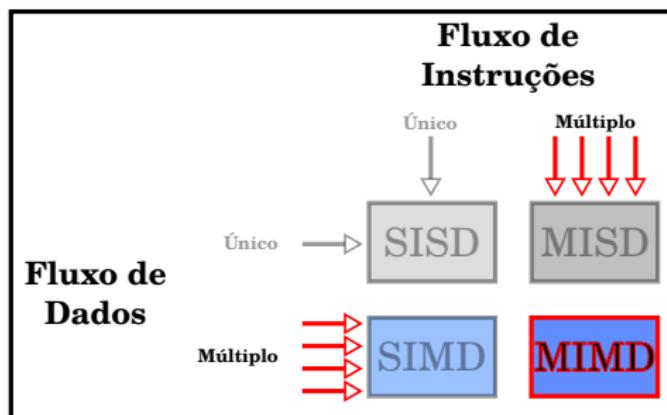


Figura : *Multiple Instruction, Multiple DAta*

Taxonomia de Flynn

Modelo para classificação de arquiteturas mais aceito pela comunidade científica [5].

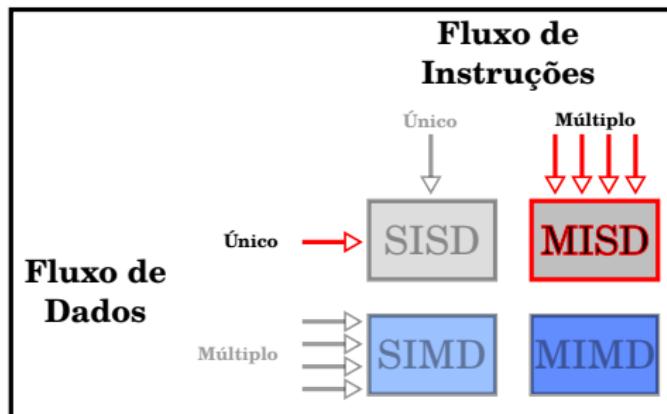


Figura : *Multiple Instruction, Single Data*

Computação Paralela

Arquiteturas Paralelas

São arquiteturas compostas por mais de um processador e que compartilham um espaço de memória em comum.

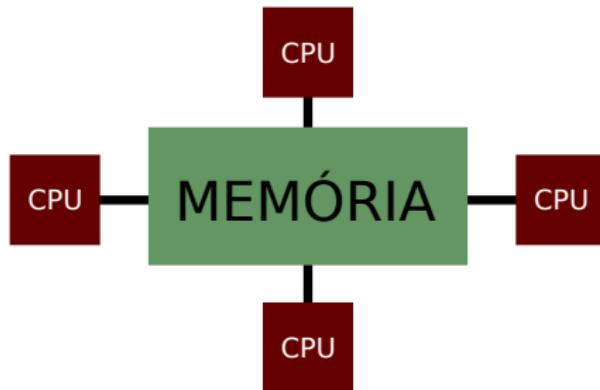
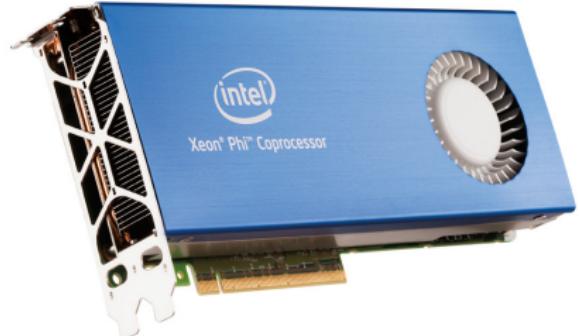


Figura : Arquiteturas Paralelas possuem uma memória compartilhada

Exemplo de Processadores Paralelos

Arquitetura *Manycore*

Utilizam técnicas de paralelismo massivo para priorizar vazão de processamento.



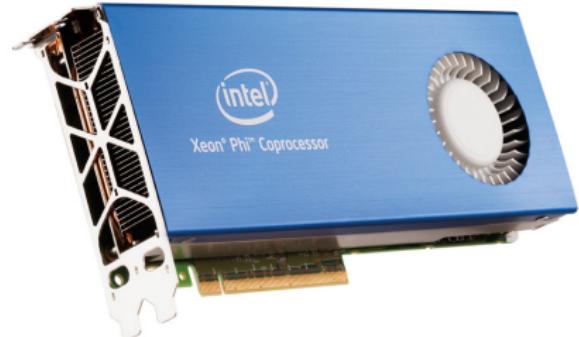
Exemplo de Processadores Paralelos

Arquitetura *Manycore*

Utilizam técnicas de paralelismo massivo para priorizar vazão de processamento.

Intel® Xeon Phi™ [6]

- 72 núcleos
- 4 threads de hardware
- 16 GB de DDR5
- Registradores vetoriais de 512 bits



Exemplo de Processadores Paralelos

Arquitetura *Manycore*

Utilizam técnicas de paralelismo massivo para priorizar vazão de processamento.

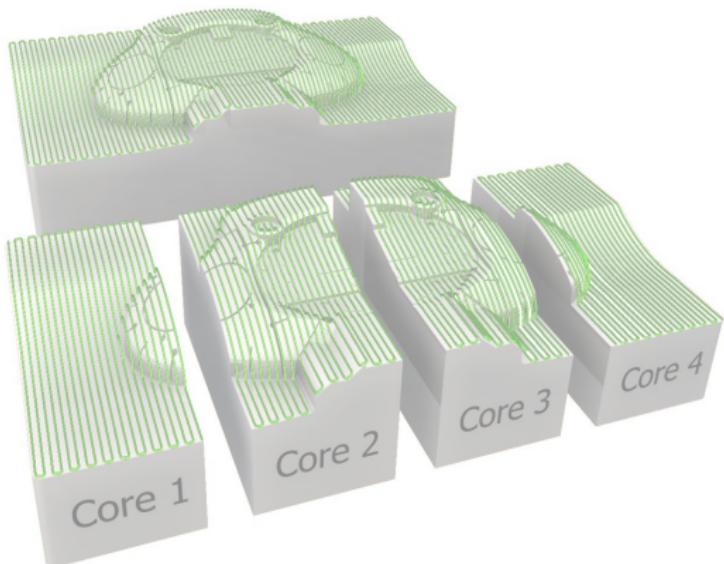


NVIDIA GeForce GTX TITAN X [7]

- 3584 CUDA cores
- 12 GB de DDR5
- Programação em CUDA

Motivação para o Uso de Computadores Paralelos

- Custo/Benefício
- Desempenho
- Disponibilidade
- Diversos outros...



Sistemas Distribuídos

Definição segundo Tanenbaum [8]

Um sistema distribuído é uma coleção de computadores autônomos conectados por uma rede de comunicação que é **percebida pelos usuários como um único computador** que provê um serviço ou resolve um problema.

Definição segundo Coulouris [9]

Um sistema distribuído é composto por computadores conectados em rede (hardware e software) que se comunicam e coordenam suas ações somente **através do envio de mensagens**.

Arquitetura de um Sistemas Distribuído

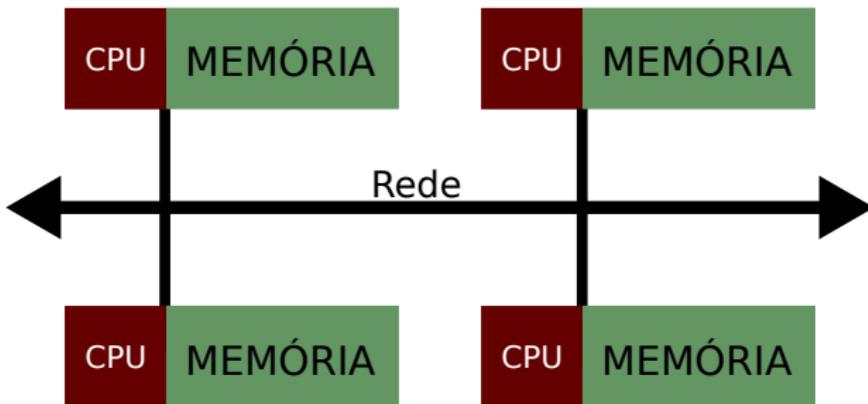
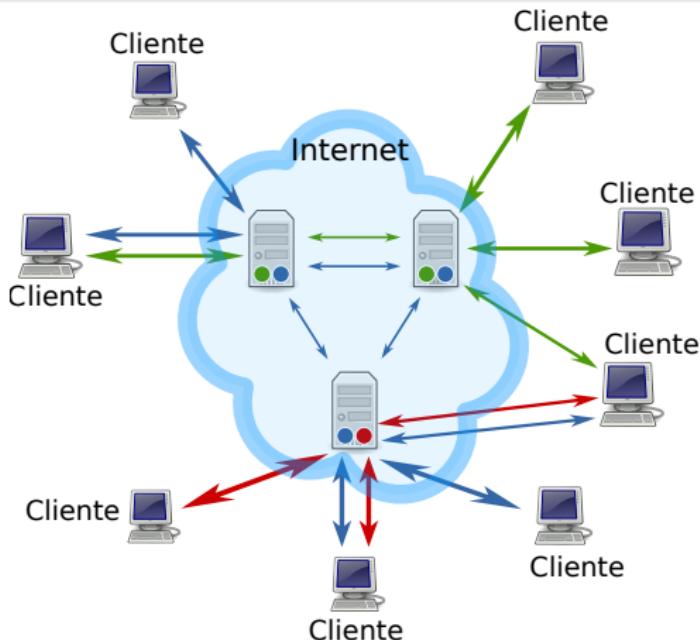


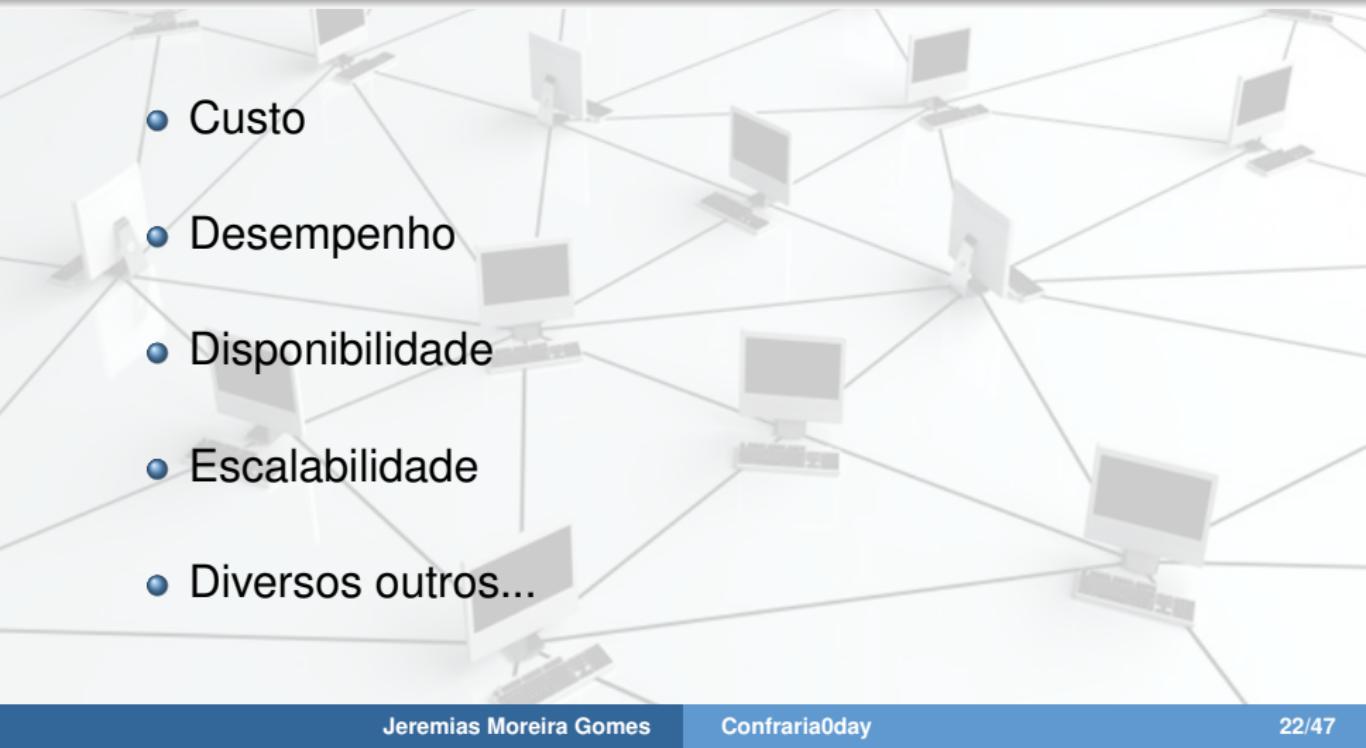
Figura : Arquitetura de um Sistema Distribuído

Exemplos de Sistemas Distribuídos

Internet



Motivação para o Uso de Sistemas Distribuídos

- 
- Custo
 - Desempenho
 - Disponibilidade
 - Escalabilidade
 - Diversos outros...

Diferença entre Sistemas Paralelos e Distribuídos

A principal diferença entre Sistemas Paralelos e Sistemas Distribuídos encontra-se apenas no objetivo de cada um.

- Sistema Paralelo: foco principal no desempenho
- Sistema Distribuído: custo, disponibilidade, etc

Tipos de Arquitetura para Computação de Alto Desempenho

Categorias

- *High Performance Computing (HPC)*
Para uso dedicado
- *High Throughput Computing (HTC)*
Aproveita a ociosidade das máquinas

High Performance Computing (HPC)

- Cluster: Arquiteturas construídas com componentes comuns (PC, etc)
- Supercomputadores: Arquiteturas proprietárias e específicas



Exemplo de Cluster

Lista TOP500

O TOP500 é um site (<http://www.top500.org>) que mostra os 500 computadores mais poderosos comercialmente conhecidos.

Características

- Divulgado duas vezes ao ano
- É de interesse de fabricantes e compradores
- Utiliza um *benchmark* chamado LINPACK
- Utiliza Tera FLOPS como unidade de medida

Ranking TOP500

Rank	Sítio	País	Sistema	Cores	TFlop/s
1	National Supercomputing Center in Wuxi	China	Sunway TaihuLight - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway,NRCPC	10.649.600	93.014
2	National Super Computer Center in Guangzhou	China	Tianhe-2 (MilkyWay-2) - TH-IVB-FEP Cluter, Intel Xeon E5-2692 12C 2.2GHz, TH Express-2, Intel Xeon Phi 31S1P, NUDT	3.120.000	33.862
3	DOE/SC/Oak Ridge National Laboratory	Estados Unidos	Titan - Cray XK7 , Opteron 6274 16C 2.200GHz, CrayGemini interconnect, NVIDIA K20x,Cray Inc	560.640	27.112
4	DOE/NNSA/LLNL	Estados Unidos	Sequoia - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom,IBM	1.572.864	20.132
5	RIKEN Advanced Institute for Computational Science (AICS)	Japão	Mira - BlueGene/Q, Power BQC 16C 1.60GHz, Custom, IBM	786.432	8.586
...
265	Laboratório Nacional de Computação Científica	Brasil	Santos Dumont GPU - Bullx B710, Intel Xeon E5-2695v2 12C 2.4GHz, Infiniband FDR, Nvidia K40	10.692	456.8
323	SENAI CIMATEC	Brasil	CIMATEC Yemoja - SGI ICE X, Intel Xeon E5-2690v2 10C 3GHz, Infiniband FDR,SGI	17.200	405.4

High Throughput Computing (HTC)

HPT

Considera a ociosidade das máquinas

- P2P - Cada usuário disponibiliza seu poder computacional
- Grid - Cria organizações virtuais a partir de empresas reais

High Throughput Computing (HTC)

HPT

Considera a ociosidade das máquinas

- P2P - Cada usuário disponibiliza seu poder computacional
- Grid - Cria organizações virtuais a partir de empresas reais



Figura : Análise de radiofrequência.

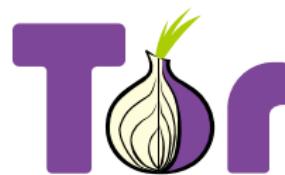
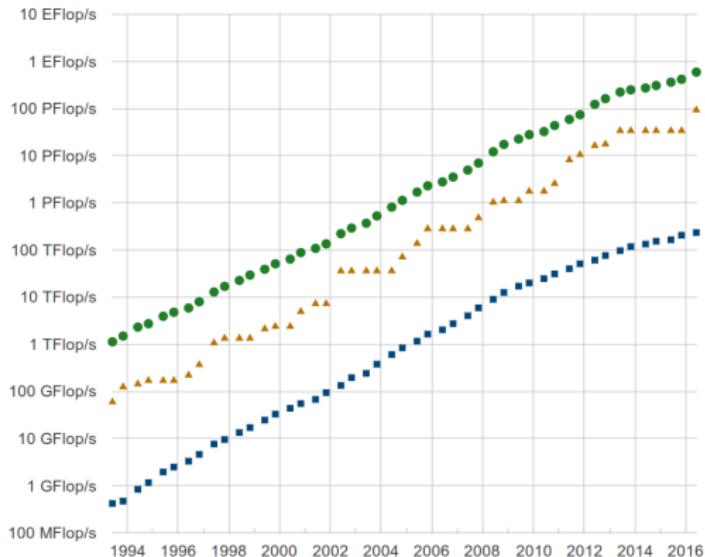


Figura : Anonimato da navegação na internet.

Quão Rápido Estamos?

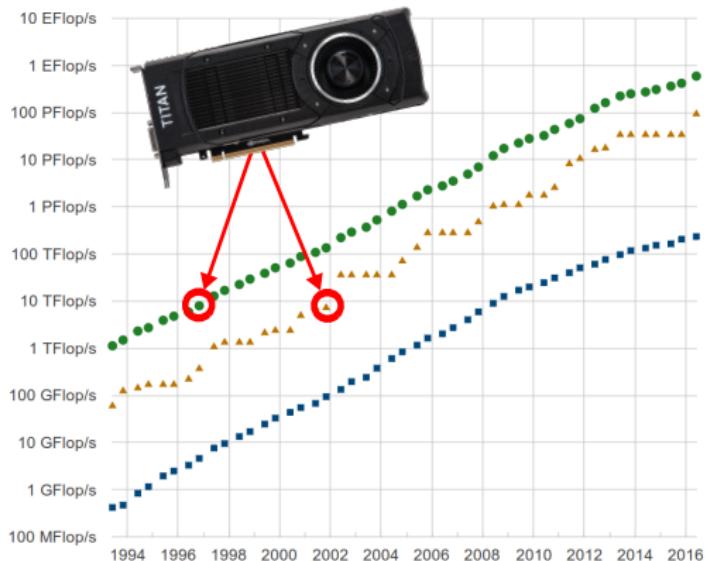
Performance Development



- A mudança a cada seis meses é extremamente dinâmica
- GTX Titan X alcança 6 Giga FLOPS

Quão Rápido Estamos?

Performance Development



- A mudança a cada seis meses é extremamente dinâmica
- GTX Titan X alcança 6 Giga FLOPS

Como Programar?

Memória Compartilhada

- Posix Threads (PThreads)
- OpenMP
- CUDA (GPU)
- OpenCL



Limites da Paralelização

Quanto meu programa ficou mais rápido?

$$S = \frac{T(1)}{T(N)}$$

$S \rightarrow$ Speedup [10]

$T(1) \rightarrow$ Tempo serial

$T(N) \rightarrow$ Tempo paralelo

Exemplo

- Tempo serial = 16 segundos
- Tempo paralelo = 3.2 segundos
- Qual o speedup?

Limites da Paralelização

Quanto meu programa ficou mais rápido?

$$S = \frac{T(1)}{T(N)}$$

$S \rightarrow$ Speedup [10]

$T(1) \rightarrow$ Tempo serial

$T(N) \rightarrow$ Tempo paralelo

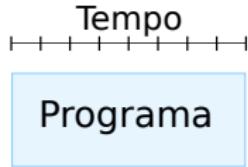
Exemplo

- Tempo serial = 16 segundos
- Tempo paralelo = 3.2 segundos
- Qual o speedup?

$$S = \frac{T(1)}{T(N)} = \frac{16}{3.2} = 5\times$$

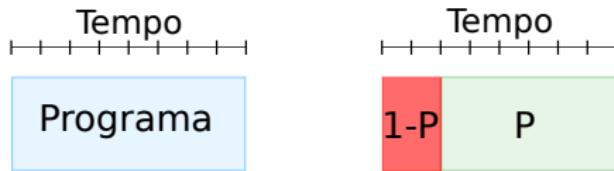
Limites da Paralelização

- Existem elementos no código que não são paralelizáveis



Limites da Paralelização

- Existem elementos no código que não são paralelizáveis

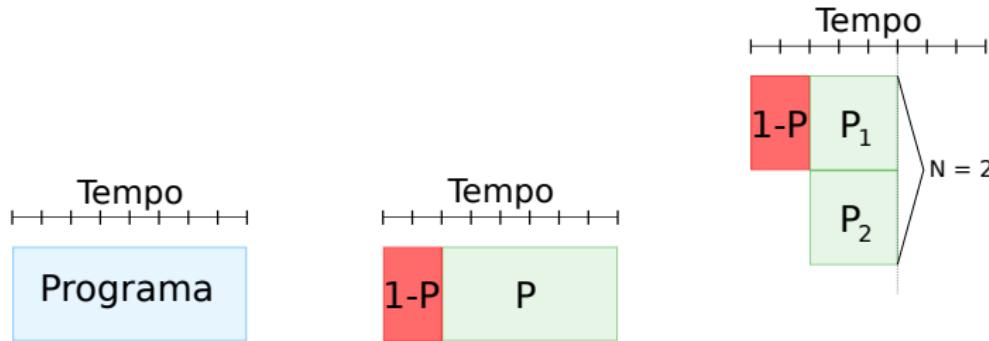


P: parte do código paralelizável

1 - P: parte do código não paralelizável

Limites da Paralelização

- Existem elementos no código que não são paralelizáveis



P: parte do código paralelizável

1 - P: parte do código não paralelizável

N: número de partições do código paralelizável

Limites da Paralelização

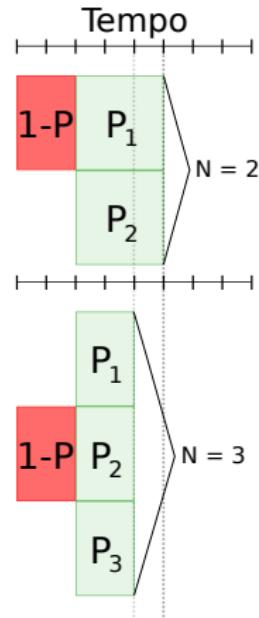
- Existem elementos no código que não são paralelizáveis



P: parte do código paralelizável

1 - P: parte do código não paralelizável

N: número de partições do código paralelizável



Exemplo 01 - Parser Paralelo

Parser de tabelas

- *Dump* de tabelas *Border Gateway Protocol* (BGP¹) [11]
- Associação de prefixos de rede
- O *Parser* auxilia na população de uma estrutura de dados para encaminhamento de pacotes (via software), utilizando *Longest Prefix Matching*

Tabela : Exemplo de Tabela de Encaminhamento.

Prefixo	Próximo Salto
192.111.0.0/18	D
192.111.42.0/22	B

Exemplo 01 - Parser Paralelo

Exemplo de *dump*

```
>1.0.0.0/24 80.241.176.31 20771 47872
1.0.4.0/24 195.208.112.161 3277 3267 6939 4637 1221 38803 56203
1.0.4.0/24 87.121.64.4 57463 5580 4637 1221 38803 56203
0
1.0.4.0/24 213.144.128.203 13030 1299 1239 4637 4637 4637 4637 38803 56203
1.0.4.0/24 198.129.33.85 293 6939 4637 1221 38803 56203
1.0.4.0/24 0
...
```

Exemplo 01 - Parser Paralelo

Exemplo de *dump*

```
>1.0.0.0/24 80.241.176.31 20771 47872
1.0.4.0/24 195.208.112.161 3277 3267 6939 4637 1221 38803 56203
1.0.4.0/24 87.121.64.4 57463 5580 4637 1221 38803 56203
0
1.0.4.0/24 213.144.128.203 13030 1299 1239 4637 4637 4637 4637 38803 56203
1.0.4.0/24 198.129.33.85 293 6939 4637 1221 38803 56203
1.0.4.0/24 0
...
...
```

Formato almejado

```
1.0.0.0/24 80.241.176.31
1.0.4.0/24 195.208.112.161
1.0.4.0/24 87.121.64.4
1.0.4.0/24 213.144.128.203
1.0.4.0/24 198.129.33.85
...
...
```

Exemplo 01 - *Parser Paralelo*

Pegando o número de processadores

```
3 file=bgptable.txt
4 cores=$(cat /proc/cpuinfo | awk '/^processor/{print $3}' | tail -1)
```

Exemplo 01 - *Parser Paralelo*

Pegando o número de processadores

```
3 file=bgptable.txt
4 cores=$(cat /proc/cpuinfo | awk '/^processor/{print $3}' | tail -1)
```

Particionamento do arquivo

```
21 $(split -n $cores $file $file.split)
22 lista='ls $file.split*'
```

Exemplo 01 - *Parser Paralelo*

Pegando o número de processadores

```
3 file=bgptable.txt
4 cores=$(cat /proc/cpuinfo | awk '/^processor/{print $3}' | tail -1)
```

Particionamento do arquivo

```
21 $(split -n $cores $file $file.split)
22 lista='ls $file.split*'
```

Aplicando o paralelismo

```
30 for i in $lista[@]
31 do
32 ( sed -e 's/^> \t0]*//' $i > $i.tmp ) &
33 done
```

Exemplo 01 - *Parser Paralelo*

Tempos de Execução

Tempo antes

```
real    26m40.517s
user    21m47.803s
sys     9m51.161s
```

Tempo depois

```
real    5m24.401s
user    40m54.087s
sys     42m52.347s
```

Exemplo 01 - Parser Paralelo

Tempos de Execução

Tempo antes

real 26m40.517s
user 21m47.803s
sys 9m51.161s

Tempo depois

real 5m24.401s
user 40m54.087s
sys 42m52.347s

Speedup

$$S = \frac{T(1)}{T(N)} = \frac{26m40s}{5m24s} = \frac{1600}{324} = 4.938\times$$

Exemplo 02 - *Capture the Flag (CTF)*

Challenge

Servidor: *localhost* Porta: 1337

Exemplo 02 - *Capture the Flag (CTF)*

Challenge

Servidor: *localhost* Porta: 1337

```
jeremias@tardis:~/Códigos/confraria0day/exemplo-02$ nc localhost 1337
```

Exemplo 02 - *Capture the Flag (CTF)*

Challenge

Servidor: *localhost* Porta: 1337

```
jeremias@tardis:~/Códigos/confraria0day/exemplo-02$ nc localhost 1337
```

+++ Confraria0day +++

- [*] Pronto para o desafio?
- [+] Para cada uma das perguntas, você tem 30 segundos para responder cada pergunta, ou a conexão será encerrada.
- [+] Para iniciar, digite o número 42:

Exemplo 02 - *Capture the Flag (CTF)*

Challenge

Servidor: *localhost* Porta: 1337

```
jeremias@tardis:~/Códigos/confraria0day/exemplo-02$ nc localhost 1337
```

+++ Confraria0day +++

- [*] Pronto para o desafio?
- [+] Para cada uma das perguntas, você tem **30 segundos** para responder cada pergunta, ou a conexão será encerrada.
- [+] Para iniciar, digite o número 42:

Exemplo 02 - *Capture the Flag (CTF)*

Challenge

Servidor: *localhost* Porta: 1337

```
jeremias@tardis:~/Códigos/confraria0day/exemplo-02$ nc localhost 1337
```

+++ Confraria0day +++

- [*] Pronto para o desafio?
- [+] Para cada uma das perguntas, você tem **30 segundos** para responder cada pergunta, ou a conexão será encerrada.
- [+] Para iniciar, digite o número 42: 42
OK, iniciando!

Exemplo 02 - *Capture the Flag* (CTF)

Challenge

Servidor: *localhost* Porta: 1337

```
jeremias@tardis:~/Códigos/confraria0day/exemplo-02$ nc localhost 1337
```

+++ Confraria0day +++

- [*] Pronto para o desafio?
 - [+] Para cada uma das perguntas, você tem **30 segundos** para responder cada pergunta, ou a conexão será encerrada.
 - [+] Para iniciar, digite o número 42: 42
OK, iniciando!
 - [+] `SHA256(X + "bWAAaPTsKhwx18jFMNS80U").hexdigest() == "d2c273208fdc3c5e1168c44..."`
X é uma string alfanumérica e $|X| == 4$
Entre com a resposta para X:

Exemplo 02 - *Capture the Flag* (CTF)

Challenge

Servidor: *localhost* Porta: 1337

```
jeremias@tardis:~/Codigos/confraria0day/exemplo-02$ nc localhost 1337
```

+++ Confraria0day +++

- [*] Pronto para o desafio?
 - [+] Para cada uma das perguntas, você tem **30 segundos** para responder cada pergunta, ou a conexão será encerrada.
 - [+] Para iniciar, digite o número 42:
OK, iniciando!
 - [+] **SHA256(X + "bWAAaPTsKhwx18jFMNS80U").hexdigest() == "d2c273208fdc3c5e1168c44..."**
X é uma string alfanumérica e **|X| == 4**
Entre com a resposta para X:

Exemplo 02 - *Capture the Flag* (CTF)

Informações Relevantes

- *SHA256*: Função de *hash* da família SHA-2[12]
- *hexdigest*: Função da biblioteca *hashlib* (python) para gerar uma string com, unicamente, valores hexadecimais
- $|X| == 4$: X possui quatro caracteres.

Exemplo 02 - *Capture the Flag* (CTF)

Informações Relevantes

- *SHA256*: Função de *hash* da família SHA-2[12]
- *hexdigest*: Função da biblioteca *hashlib* (python) para gerar uma string com, unicamente, valores hexadecimais
- $|X| == 4$: X possui quatro caracteres.
Mas quais caracteres?

ABCDEFGHIJKLMNPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789 \$%*+-./:



71 caracteres diferentes (incluindo o espaço)

Exemplo 02 - *Capture the Flag* (CTF)

Objetivo

Dado o pedaço da string de entrada, eu preciso achar quais são os quatro caracteres restantes geram o *hash* completo.

_____ bWAAaPTsKhwx18jFMNS80U

Exemplo 02 - *Capture the Flag* (CTF)

Objetivo

Dado o pedaço da string de entrada, eu preciso achar quais são os quatro caracteres restantes geram o *hash* completo.

_____ bWAAaPTsKhwx18jFMNS80U

A A A A

A A A B

A A A C

...

Exemplo 02 - *Capture the Flag* (CTF)

Objetivo

Dado o pedaço da string de entrada, eu preciso achar quais são os quatro caracteres restantes geram o *hash* completo.

_____ bWAAaPTsKhwx18jFMNS80U

A A A A
A A A B
A A A C
...
: : : :

$$A(71, 4) = \frac{71!}{(71-4)!} = \frac{71*70*69*68*67!}{67!} = 23.662.170$$

Se 1.000.000 operações, levam em média 1 segundo, então se cada *hash* gastasse 1 operação, levariam em média $\sim 24s$ o pior caso. Testes apontaram $\sim 68s$.

Exemplo 02 - *Capture the Flag* (CTF)

Solução



Exemplo 02 - *Capture the Flag* (CTF)

Estratégia

A A : A : A :

B A : A : A :

.

Q A : A : A :

R A : A : A :

T A : A : A :

.

h A : A : A :

Exemplo 02 - *Capture the Flag* (CTF)

Estratégia

A A : A : A :

B A : A : A :

.

Q A : A : A :

R A : A : A :

T A : A : A :

.

h A : A : A :

Exemplo 02 - *Capture the Flag* (CTF)

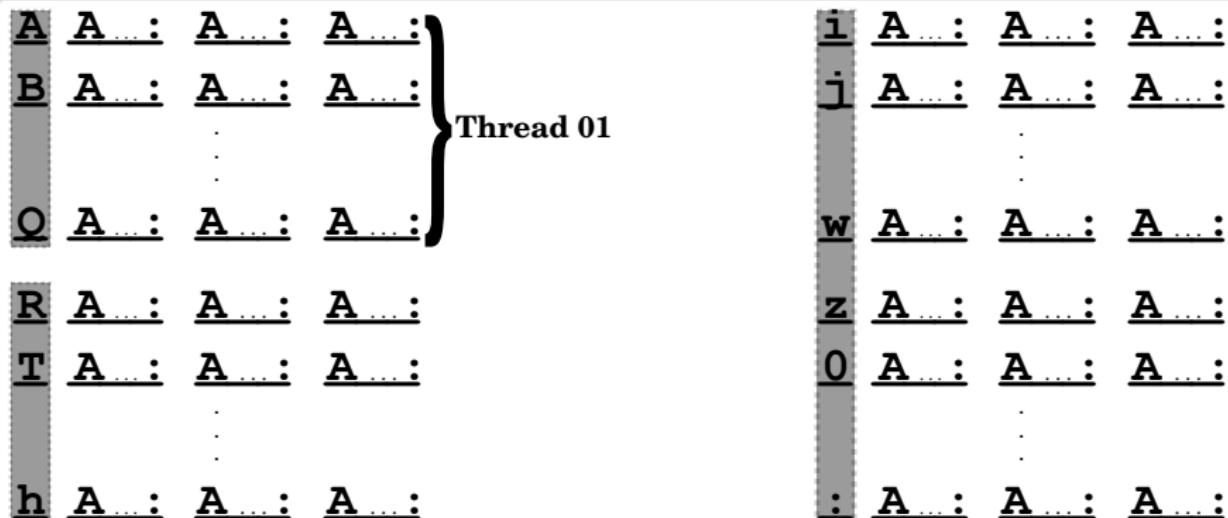
Estratégia

A A : A : A :
B A : A : A :
C A : A : A :
D A : A : A :
E A : A : A :
F A : A : A :
G A : A : A :
H A : A : A :

i A : A : A :
j A : A : A :
k A : A : A :
l A : A : A :
m A : A : A :
n A : A : A :
o A : A : A :
p A : A : A :
q A : A : A :
r A : A : A :
s A : A : A :
t A : A : A :
u A : A : A :
v A : A : A :
w A : A : A :
x A : A : A :
y A : A : A :
z A : A : A :
0 A : A : A :
1 A : A : A :
2 A : A : A :
3 A : A : A :
4 A : A : A :
5 A : A : A :
6 A : A : A :
7 A : A : A :
8 A : A : A :
9 A : A : A :
: A : A : A :

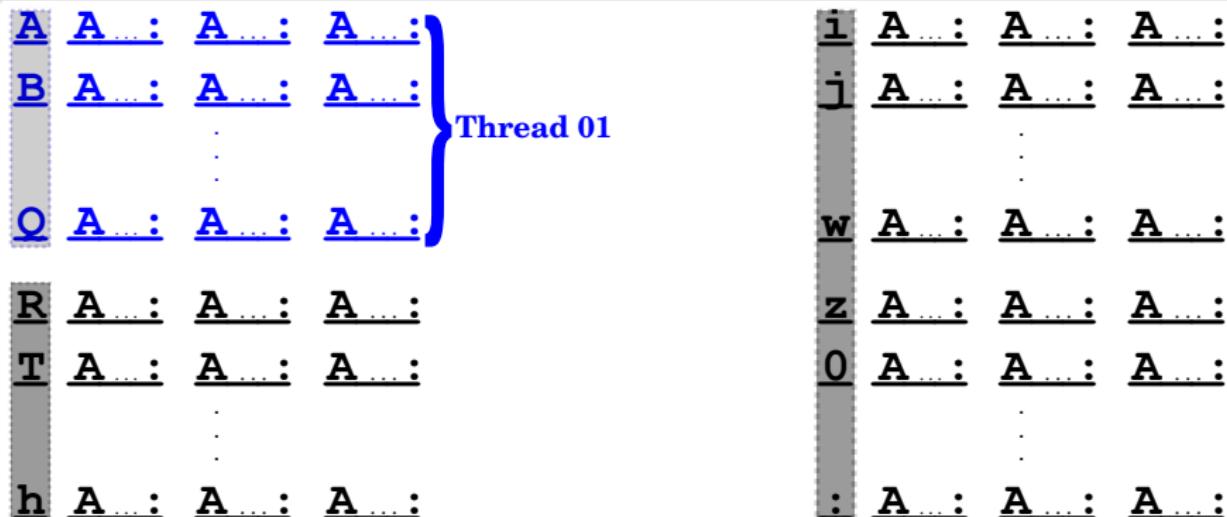
Exemplo 02 - *Capture the Flag* (CTF)

Estratégia



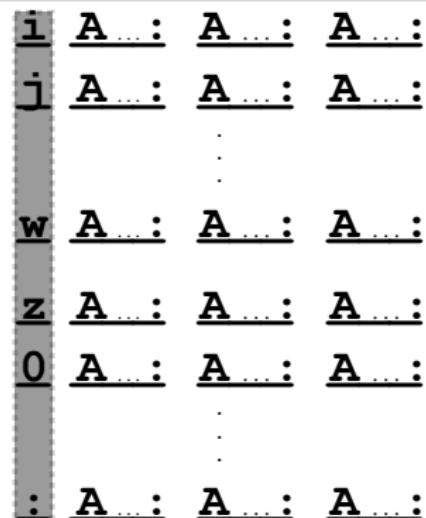
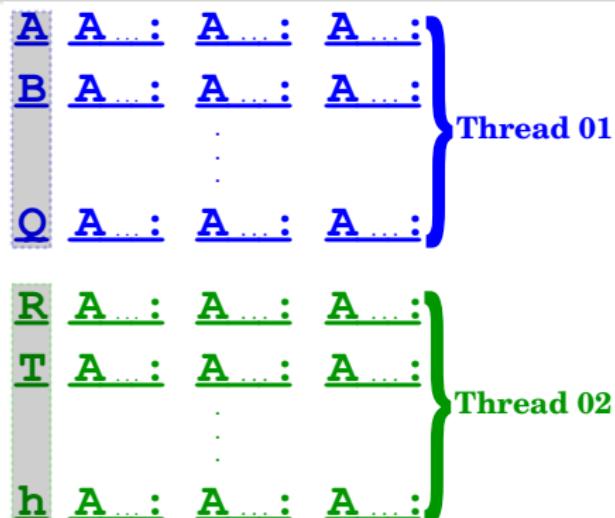
Exemplo 02 - *Capture the Flag* (CTF)

Estratégia



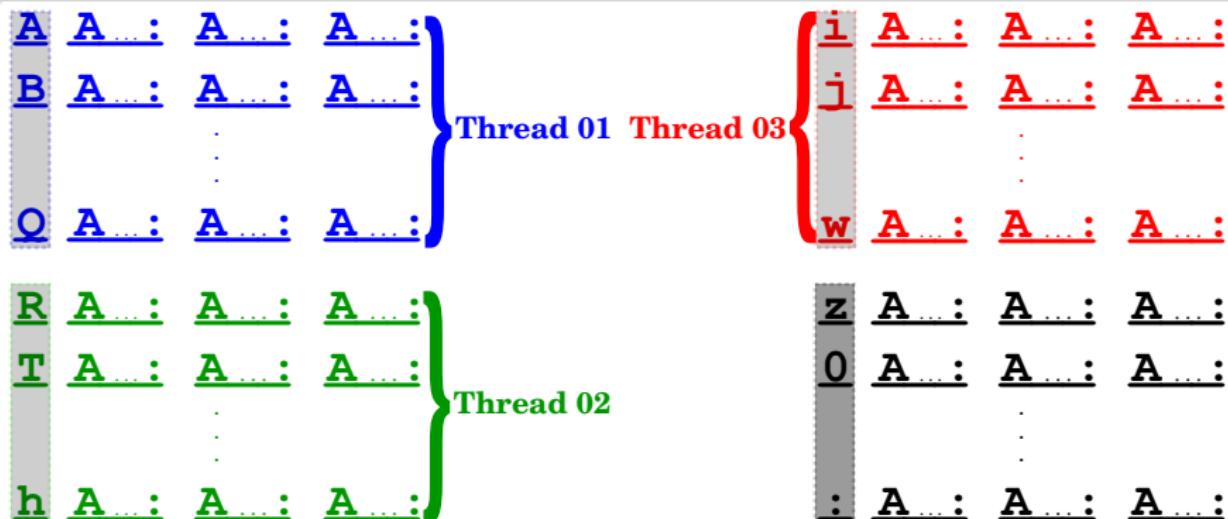
Exemplo 02 - *Capture the Flag* (CTF)

Estratégia



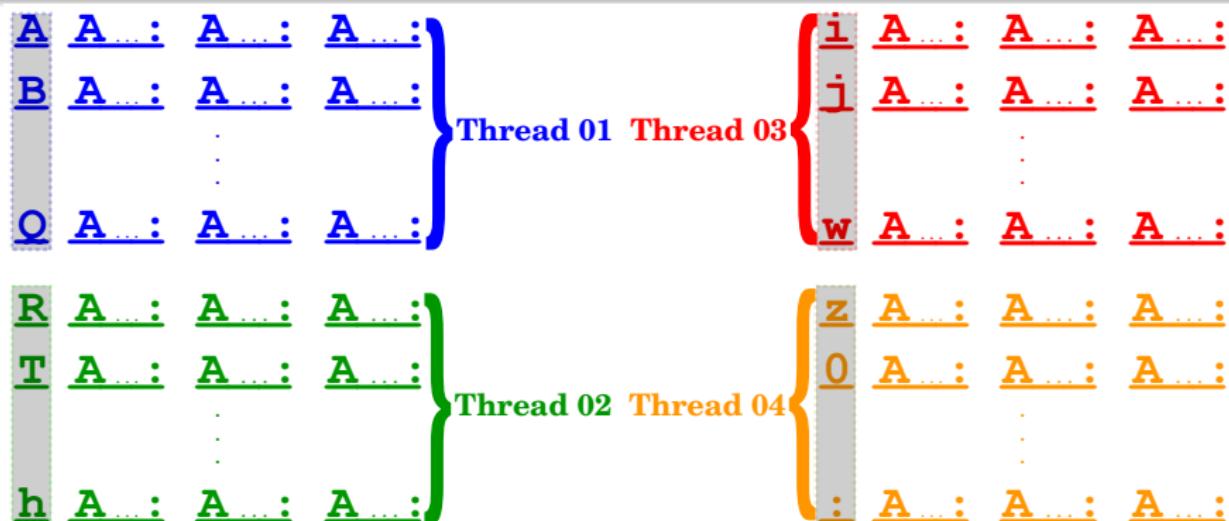
Exemplo 02 - *Capture the Flag* (CTF)

Estratégia



Exemplo 02 - *Capture the Flag* (CTF)

Estratégia



Exemplo 02 - *Capture the Flag* (CTF)

Trecho de Código

```
01 #pragma omp parallel for firstprivate(j, k, l, guess, hash)
02 for (i = 0; i < 71; i++) {
03     guess[0] = an[i];
04     for (j = 0; j < 71; j++) {
05         guess[1] = an[j];
06         for (k = 0; k < 71; k++) {
07             guess[2] = an[k];
08             for (l = 0; l < 71; l++) {
09                 guess[3] = an[l];
10                 genSHA256(guess, hash);
11                 if (strstr(hash, target)) {
12                     for (int ii = 0; ii < 4; ii++) {
13                         buffer[ii] = guess[ii];
14                     }
15                     buffer[4] = '\0';
16                     #pragma omp cancel for
17                 }
18             }
19         }
20     }
21 #pragma omp cancellation point for
22 }
```

Exemplo 02 - *Capture the Flag* (CTF)

Trecho de Código

```
01 #pragma omp parallel for firstprivate(j, k, l, guess, hash)
02 for (i = 0; i < 71; i++) {
03     guess[0] = an[i];
04     for (j = 0; j < 71; j++) {
05         guess[1] = an[j];
06         for (k = 0; k < 71; k++) {
07             guess[2] = an[k];
08             for (l = 0; l < 71; l++) {
09                 guess[3] = an[l];
10                 genSHA256(guess, hash);
11                 if (strstr(hash, target)) {
12                     for (int ii = 0; ii < 4; ii++) {
13                         buffer[ii] = guess[ii];
14                     }
15                     buffer[4] = '\0';
16                     #pragma omp cancel for
17                 }
18             }
19         }
20     }
21 #pragma omp cancellation point for
22 }
```

- ➊ `#pragma omp parallel for`
 - Paraleliza o *loop* mais externo

Exemplo 02 - *Capture the Flag* (CTF)

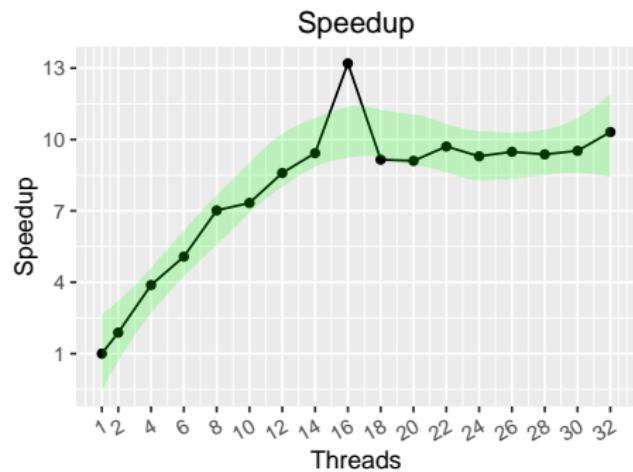
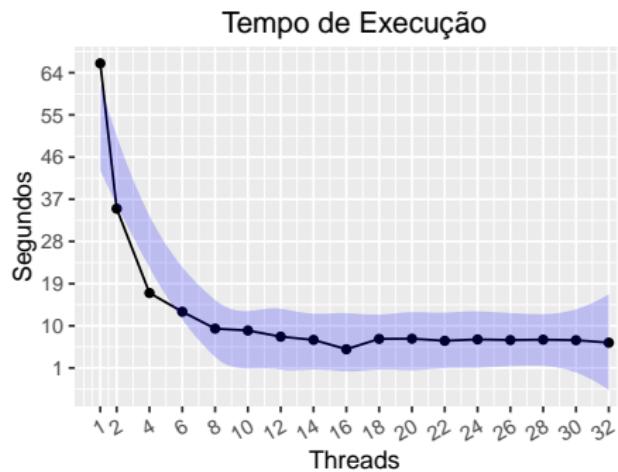
Trecho de Código

```
01 #pragma omp parallel for firstprivate(j, k, l, guess, hash)
02 for (i = 0; i < 71; i++) {
03     guess[0] = an[i];
04     for (j = 0; j < 71; j++) {
05         guess[1] = an[j];
06         for (k = 0; k < 71; k++) {
07             guess[2] = an[k];
08             for (l = 0; l < 71; l++) {
09                 guess[3] = an[l];
10                 genSHA256(guess, hash);
11                 if (strstr(hash, target)) {
12                     for (int ii = 0; ii < 4; ii++) {
13                         buffer[ii] = guess[ii];
14                     }
15                     buffer[4] = '\0';
16                     #pragma omp cancel for
17                 }
18             }
19         }
20     }
21 #pragma omp cancellation point for
22 }
```

- **#pragma omp parallel for**
 - Paraleliza o *loop* mais externo
- **#pragma omp cancel for**
 - *Break*, caso a string seja encontrada
- **#pragma omp cancellation point for**
 - Outras *threads* veem um *break*

Exemplo 02 - *Capture the Flag* (CTF)

Testes de Tempo



Testes de Tempo e Speedup onde a flag é sempre “:::”

Conclusão

- Limite físico
- E se ocorrer um salto de performance?

Referências Bibliográficas I

-  I. H. P. Computing, "Inside high performance computing site."
<http://insidehpc.com/hpc-basic-training/what-is-hpc/>, 2015.
-  G. E. Moore, "Cramming More Components onto Integrated Circuits, Reprinted from Electronics, volume 38, number 8, April 19, 1965, pp. 114 ff.," *IEEE Solid-State Circuits Newsletter*, vol. 3, no. 20, pp. 33–35, 2006.
-  L. I. Millett, S. H. Fuller, *et al.*, *The Future of Computing Performance:: Game Over or Next Level?* National Academies Press, 2011.
-  Top500, "Top500 supercomputer sites." <http://top500.org>, 2015.
-  M. Flynn, "Some Computer Organizations and their Effectiveness," *Computers, IEEE Transactions on*, vol. 100, no. 9, pp. 948–960, 1972.
-  I. Corporation, "Intel xeon phi." <http://ark.intel.com/pt-br/products/95830>, 2016.
-  N. Corporation, "Nvidia titan x." <http://www.nvidia.com.br/object/geforce-gtx-titan-x-br.html>, 2016.
-  A. S. Tanenbaum, *Distributed operating systems*. Pearson Education India, 1995.

Referências Bibliográficas II

-  G. F. Coulouris, J. Dollimore, and T. Kindberg, *Distributed systems: concepts and design*. Pearson Education, 2005.
-  G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," in *Proceedings of the April 18-20, 1967, spring joint computer conference*, pp. 483–485, ACM, 1967.
-  B. R. T. A. Reports, "Bgp routing table analysis reports." <http://bgp.potaroo.net/>, 2016.
-  J. Katz and Y. Lindell, *Introduction to modern cryptography*. CRC Press, 2014.

A Necessidade da Computação de Alto Desempenho para os Dias Atuais

Limites da Paralelização

Jeremias Moreira Gomes

jeremiasmg@gmail.com

29 de setembro de 2016