

Introdução

MCZA020-13 - Programação Paralela

Emilio Francesquini

e.francesquini@ufabc.edu.br

2025.Q2

Centro de Matemática, Computação e Cognição
Universidade Federal do ABC



- Estes slides foram preparados para o curso de **Programação Paralela na UFABC**.
- Este material pode ser usado livremente desde que sejam mantidos, além deste aviso, os créditos aos autores e instituições.

Programação Paralela

- De 1986 até 2002 o desempenho de processadores aumentou em média **50% ao ano**
- Desde 2002 o desempenho de (um único núcleo) está por volta de **20% ao ano**
- Não parece muita diferença?

- De 1986 até 2002 o desempenho de processadores aumentou em média **50% ao ano**
- Desde 2002 o desempenho de (um único núcleo) está por volta de **20% ao ano**
- Não parece muita diferença?
 - ▶ **60x** em 10 anos vs. **6x**
- Limites do desenvolvimento levaram aos **multi-core**

- 1 E daí que agora “só” melhora 20% ao ano? Já não é rápido o suficiente?
- 2 Porque os projetistas não fabricam mais processadores com melhoras tão significativas de desempenho? Por que eles estão cada vez mais construindo **sistemas paralelos**?
- 3 Porque não automatizamos a transformação dos atuais programas sequenciais em **programas paralelos**? Por que precisamos escrever programas paralelos?

Pergunta 1: E daí que são apenas 20%?

Algumas áreas que se beneficiariam enormemente de um aumento considerável na capacidade de processamento:

- Modelos climáticos
- Dobramento de proteínas
- Desenvolvimento de fármacos
- Pesquisas energéticas
- Análise de dados
- Inteligência Artificial

■ Características

- ▶ Definitivamente *data-intensive*
- ▶ Mas podem também ser *processing-intensive*

■ Exemplos:

- ▶ *Crawling*, indexação, busca, mineração de dados da web e IA
- ▶ Pesquisa em biologia computacional na era “pós-genômica”
- ▶ Processamento de dados científicos (física, astronomia, etc.)
- ▶ Computação Verde
- ▶ etc.

Problemas da ordem de **petabytes**!

$$\begin{aligned} 1 \text{ PB} &= 1.000.000.000.000.000 \text{ B} \\ &= 1.000^5 \text{ B} \\ &= 10^{15} \text{ B} \\ &= 1 \text{ milhão de gigabytes} \\ &= 1 \text{ mil terabytes} \end{aligned}$$

Muitos, mas muitos dados

- O Google processa cerca de **20 petabytes** de dados por dia (Dados de 2008 - fontes mais novas difíceis de encontrar)
- O Wayback Machine tem cerca de **100 petabytes** (Nov/2024) + **100 terabytes/dia** (mar/2009)
- O Facebook tinha cerca de **2,5 petabytes de dados de usuários** + **15 terabytes/dia** (abr/2009) e construiu um ambiente de **300 petabytes** (2014).
- O site eBay tem cerca de **6,5 petabytes de dados dos usuários** + **50 terabytes/dia** (mai/2009)
- O Grande Colisor de Hádrons do CERN irá gerar cerca de **15 petabytes/ano**

Muitos, mas muitos dados

- O Google processa cerca de **20 petabytes** de dados por dia (Dados de 2008 - fontes mais novas difíceis de encontrar)
- O Wayback Machine tem cerca de **100 petabytes** (Nov/2024) + **100 terabytes/dia** (mar/2009)
- O Facebook tinha cerca de **2,5 petabytes de dados de usuários + 15 terabytes/dia** (abr/2009) e construiu um ambiente de **300 petabytes** (2014).
- O site eBay tem cerca de **6,5 petabytes de dados dos usuários + 50 terabytes/dia** (mai/2009)
- O Grande Colisor de Hádrons do CERN irá gerar cerca de **15 petabytes/ano**

Muitos, mas muitos dados

- O Google processa cerca de **20 petabytes** de dados por dia (Dados de 2008 - fontes mais novas difíceis de encontrar)
- O Wayback Machine tem cerca de **100 petabytes** (Nov/2024) + **100 terabytes/dia** (mar/2009)
- O Facebook tinha cerca de **2,5 petabytes de dados de usuários + 15 terabytes/dia** (abr/2009) e construiu um ambiente de **300 petabytes** (2014).
- O site eBay tem cerca de **6,5 petabytes de dados dos usuários + 50 terabytes/dia** (mai/2009)
- O Grande Colisor de Hádrons do CERN irá gerar cerca de **15 petabytes/ano**

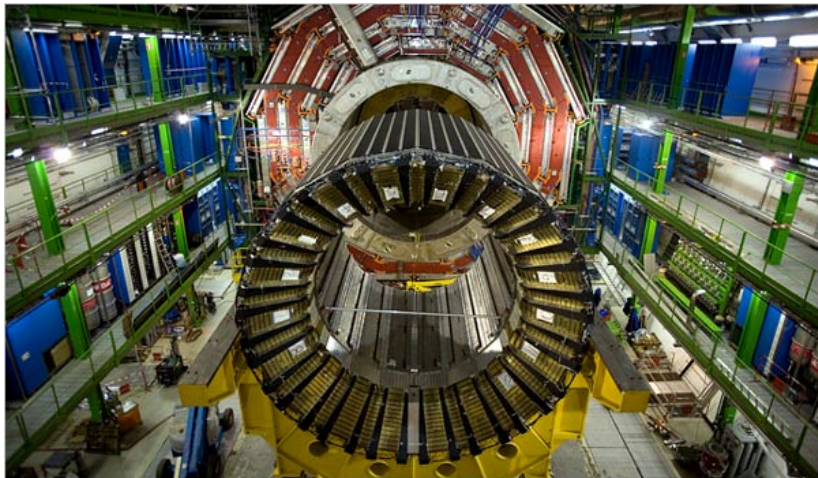
Muitos, mas muitos dados

- O Google processa cerca de **20 petabytes** de dados por dia (Dados de 2008 - fontes mais novas difíceis de encontrar)
- O Wayback Machine tem cerca de **100 petabytes** (Nov/2024) + **100 terabytes/dia** (mar/2009)
- O Facebook tinha cerca de **2,5 petabytes de dados de usuários + 15 terabytes/dia** (abr/2009) e construiu um ambiente de **300 petabytes** (2014).
- O site eBay tem cerca de **6,5 petabytes de dados dos usuários + 50 terabytes/dia** (mai/2009)
- O Grande Colisor de Hádrons do CERN irá gerar cerca de **15 petabytes/ano**

Muitos, mas muitos dados

- O Google processa cerca de **20 petabytes** de dados por dia (Dados de 2008 - fontes mais novas difíceis de encontrar)
- O Wayback Machine tem cerca de **100 petabytes** (Nov/2024) + **100 terabytes/dia** (mar/2009)
- O Facebook tinha cerca de **2,5 petabytes de dados de usuários + 15 terabytes/dia** (abr/2009) e construiu um ambiente de **300 petabytes** (2014).
- O site eBay tem cerca de **6,5 petabytes de dados dos usuários + 50 terabytes/dia** (mai/2009)
- O Grande Colisor de Hádrons do CERN irá gerar cerca de **15 petabytes/ano**

De qual volume de dados estamos falando?



$$\begin{aligned}1 \text{ PB} &= 1.000.000.000.000.000 \text{ B} \\&= 1.000^5 \text{ B} \\&= 10^{15} \text{ B} \\&= 1 \text{ milhão de gigabytes} \\&= 1 \text{ mil terabytes}\end{aligned}$$

Ou seja, os **15 petabytes** que o CERN irá gerar por ano equivalem a **15 milhões de gigabytes**. Seriam necessários **1,7 milhão de DVDs *dual-layer*** para armazenar tanta informação!

- Encontram informações sobre novos fatos
 - ▶ Casamento de padrões com informações da web
 - ▶ ex: quem matou John Lennon?
- Procuram por novas relações entre os dados
 - ▶ Alguns padrões levam a novas relações:
 - os fatos: "Nascimento-de(Mozart, 1756)" e "Nascimento-de(Einstein, 1879)"
 - levam aos dados: "Wolfgang Amadeus Mozart (1756–1791)" e "Einstein nasceu em 1879"
 - que levam a diferentes padrões: "PESSOA (DATA –)" e "PESSOA nasceu em DATA"
 - que, por sua vez, permitem encontrar novos fatos

- Encontram informações sobre novos fatos
 - ▶ Casamento de padrões com informações da web
 - ▶ ex: quem matou John Lennon?
- Procuram por novas relações entre os dados
 - ▶ Alguns padrões levam a novas relações:
 - os fatos: “Nascimento-de(Mozart, 1756)” e “Nascimento-de(Einstein, 1879)”
 - levam aos dados: “Wolfgang Amadeus Mozart (1756–1791)” e “Einstein nasceu em 1879”
 - que levam a diferentes padrões: “PESSOA (DATA –)” e “PESSOA nasceu em DATA”
 - que, por sua vez, permitem encontrar novos fatos

- Encontram informações sobre novos fatos
 - ▶ Casamento de padrões com informações da web
 - ▶ ex: quem matou John Lennon?
- Procuram por novas relações entre os dados
 - ▶ Alguns padrões levam a novas relações:
 - os fatos: “Nascimento-de(Mozart, 1756)” e “Nascimento-de(Einstein, 1879)”
 - levam aos dados: “Wolfgang Amadeus Mozart (1756–1791)” e “Einstein nasceu em 1879”
 - que levam a diferentes padrões: “PESSOA (DATA –)” e “PESSOA nasceu em DATA”
 - que, por sua vez, permitem encontrar novos fatos

- Encontram informações sobre novos fatos
 - ▶ Casamento de padrões com informações da web
 - ▶ ex: quem matou John Lennon?
- Procuram por novas relações entre os dados
 - ▶ Alguns padrões levam a novas relações:
 - os fatos: “Nascimento-de(Mozart, 1756)” e “Nascimento-de(Einstein, 1879)”
 - levam aos dados: “Wolfgang Amadeus Mozart (1756–1791)” e “Einstein nasceu em 1879”
 - que levam a diferentes padrões: “PESSOA (DATA –)” e “PESSOA nasceu em DATA”
 - que, por sua vez, permitem encontrar novos fatos

- Encontram informações sobre novos fatos
 - ▶ Casamento de padrões com informações da web
 - ▶ ex: quem matou John Lennon?
- Procuram por novas relações entre os dados
 - ▶ Alguns padrões levam a novas relações:
 - os fatos: “Nascimento-de(Mozart, 1756)” e “Nascimento-de(Einstein, 1879)”
 - levam aos dados: “Wolfgang Amadeus Mozart (1756–1791)” e “Einstein nasceu em 1879”
 - que levam a diferentes padrões: “PESSOA (DATA –)” e “PESSOA nasceu em DATA”
 - que, por sua vez, permitem encontrar novos fatos

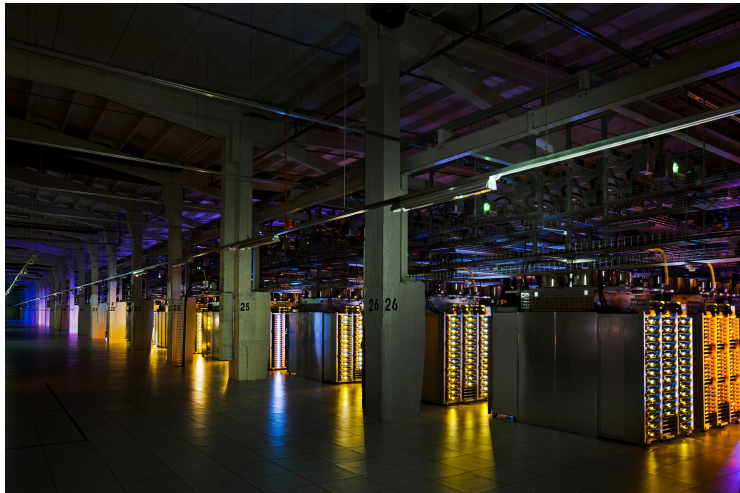
- Encontram informações sobre novos fatos
 - ▶ Casamento de padrões com informações da web
 - ▶ ex: quem matou John Lennon?
- Procuram por novas relações entre os dados
 - ▶ Alguns padrões levam a novas relações:
 - os fatos: “Nascimento-de(Mozart, 1756)” e “Nascimento-de(Einstein, 1879)”
 - levam aos dados: “Wolfgang Amadeus Mozart (1756–1791)” e “Einstein nasceu em 1879”
 - que levam a diferentes padrões: “PESSOA (DATA –)” e “PESSOA nasceu em DATA”
 - que, por sua vez, permitem encontrar novos fatos

Pergunta:

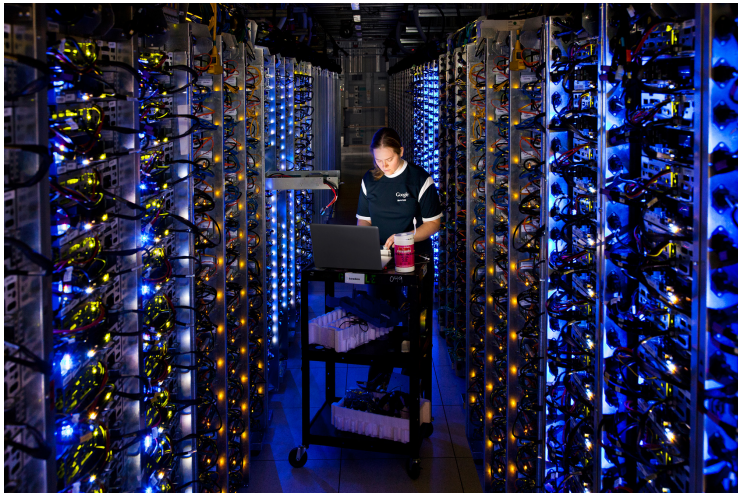
Quão grandes são os *data centers* que fazem sistemas que afetam a vida de quase todo mundo que se conecta a Internet (como os do Google, Facebook, etc.) funcionarem?



Fonte: <http://www.google.com/intl/pt-BR/about/datacenters/>



Fonte: <http://www.google.com/intl/pt-BR/about/datacenters/>



Fonte: <http://www.google.com/intl/pt-BR/about/datacenters/>

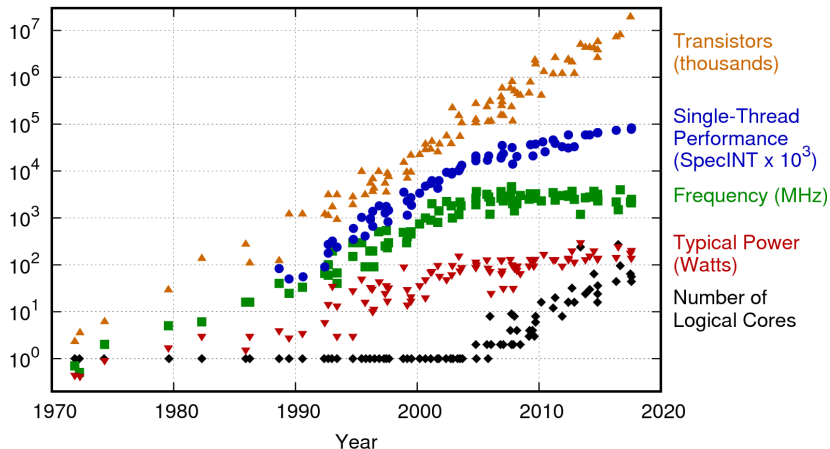


Fonte: <http://www.google.com/intl/pt-BR/about/datacenters/>

Só o Google tem 25 desses espalhados pelo mundo com mais 13 em implantação!

Pergunta 2: Por que os projetistas não fabricam mais processadores com melhoras tão significativas de desempenho? Por que eles estão cada vez mais construindo **sistemas paralelos**?

42 Years of Microprocessor Trend Data



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2017 by K. Rupp

Fonte: <https://www.karlrupp.net/2018/02/42-years-of-microprocessor-trend-data/>

- Desde a década de 80 até o início dos anos 2000 projetistas de CPUs conseguiam melhorar o desempenho dos processadores em três principais linhas:
 - ▶ Frequência de funcionamento
 - ▶ Otimização das operações
 - ▶ Caches
- Essa época acabou. Hoje desenvolvedores precisam se preocupar com o desempenho **sequencial** dos seus programas.
- Programas antigos não vão mais melhorar o seu desempenho no mesmo ritmo que antes “só na banguela”

Fonte: <http://www.gotw.ca/publications/concurrency-ddj.htm>

Pergunta 3: Porque não automatizamos a transformação dos atuais programas sequenciais em **programas paralelos**? Por que precisamos escrever programas paralelos?

- Não ajuda rodar mais de uma instância do seu jogo favorito ao mesmo tempo
 - ▶ É preciso escrever códigos que se utilizem do hardware adicional
 - ▶ É preciso escrever códigos capazes de **rodar em paralelo**
- Não é fácil encontrar (automaticamente) uma maneira de paralelizar um código.
 - ▶ Na verdade **quase sempre** fazer um programa paralelo...
 - eficiente
 - correto
 - simples
 - ▶ ...é **muito mais difícil** do que escrever uma versão sequencial com essas mesmas características
 - ▶ Se alguém te falar o contrário esta pessoa está muito provavelmente sofrendo do Efeito Dunning-Kruger (https://pt.wikipedia.org/wiki/Efeito_Dunning-Kruger)

Programa sequencial:

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

```
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;
}
```

Se tivermos, por exemplo, **8 processadores**, $n = 24$, e as chamadas para `Compute_next_value(...)` devolverem:

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

Então os valores armazenados na variável `my_sum` serão:

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

```
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;  
}
```

No nosso exemplo, se o *core* mestre for o 0, então ele somaria os valores $8 + 19 + 7 + 15 + 7 + 13 + 12 + 14 = 95$.

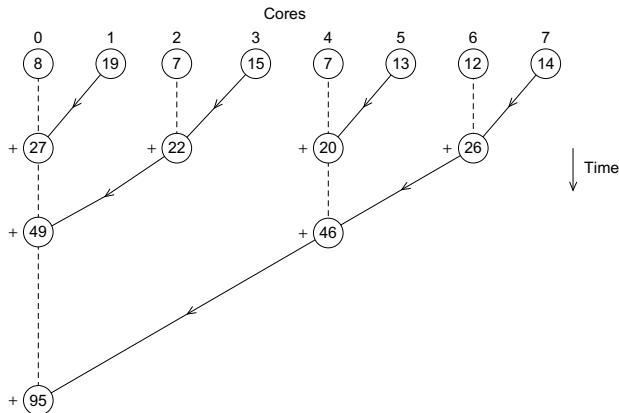
Pergunta 1:

Quantas somas o core mestre faz no caso geral? É possível melhorar?

```
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;  
}
```

Pergunta 2:

Quantas comunicações cada um dos processos faz no caso geral? É possível melhorar?



Fonte: [PP]

Pergunta:

Quantas somas e quantas comunicações cada um dos *cores* faz neste caso?

Como escrever programas paralelos

- **Paralelismo de tarefas (*task parallelism*)**
 - ▶ Divisão das tarefas a serem feitas entre os cores
- **Paralelismo de dados (*data parallelism*)**
 - ▶ Divisão dos dados a serem processados entre os cores

Exercício:

O Professor Espertalhão tem 100 alunos que acabaram de fazer uma prova com 4 questões. Ele também possui 4 monitores que podem ajudá-lo a corrigir a prova. Sugira maneiras que o Prof. Espertalhão pode usar para paralelizar a correção.

Exercício:

Classifique as características das implementações de soma que acabamos de discutir em paralelismo de dados e de tarefas.

- Alguns problemas são classificados como **vergonhosamente paralelos** (*embarrassingly parallel*)
 - ▶ Você também vai ver por aí “embaraçosamente paralelos”
- Isto significa que nenhuma (ou praticamente nenhuma) coordenação entre os processos que estão executando é necessária.
- Exemplos:
 - ▶ Converter todos os arquivos em um diretório de JPEG para PNG
 - ▶ Mandar e-mails (enviar Spam :p) para todos os e-mails listados em um arquivo
 - ▶ Converter uma imagem colorida para preto-e-branco
 - ▶ Quebra de senhas
 - ▶ Mineração de Bitcoins
 - ▶ Procurar ETs!
 - ▶ Treinar redes neurais (pelo menos boa parte!)

Pergunta:

Nosso exemplo de soma (versão 1 e 2) **não é** vergonhosamente paralelo. Por quê?

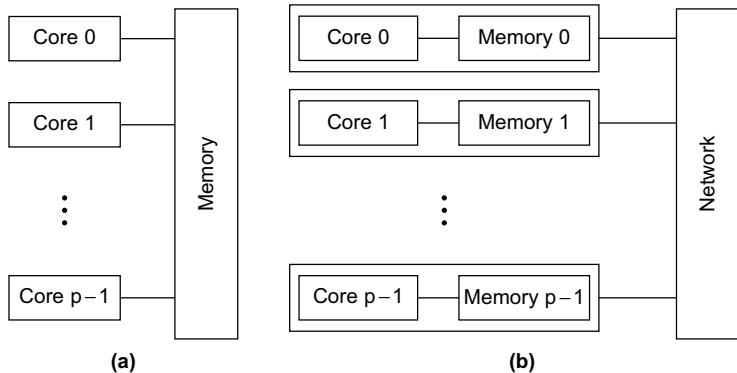
Nosso exemplo da soma emprega diversos mecanismos de coordenação:

- **Comunicação** – um ou mais cores enviam suas somas parciais para outros cores
- **Balanceamento de carga (*load balancing*)** – Apesar de termos fórmulas fechadas para o trabalho a ser desempenhado por cada core, queremos que o trabalho seja bem distribuído.
 - ▶ Minimização do **caminho crítico (*critical path*)**
- **Sincronização** – suponha que em vez de calcular os próximos valores a serem somados, estes valores fossem lidos da **stdin** pelo core mestre
 - ▶ Os demais cores **precisam esperar o core mestre acabar a leitura** (pelo menos da parte que lhes diz respeito) antes de começar a trabalhar!

Daqui até o fim do quadrimestre...

- Algumas linguagens e bibliotecas fornecem mecanismos que são capazes de (em alguns casos) fazer a paralelização automática de código.
 - ▶ Trocam desempenho por facilidade
 - ▶ Vamos comentar um pouco sobre elas neste curso
- Neste curso, contudo, não estamos interessados nessas ferramentas. Nós veremos como fazer **paralelismo explícito**.
- Utilizaremos a linguagem de programação C.

Com quais tipos de máquinas trabalharemos?



Fonte: [PP]

(a) memória compartilhada (b) memória distribuída

Pergunta:

Em qual delas roda o seu Fortnite? Em qual delas roda sua busca na Web?

- Aprenderemos as seguintes ferramentas
 - ▶ Message Passing Interface - **MPI**
 - ▶ POSIX Threads - **Pthreads**
 - ▶ **OpenMP**
 - ▶ **CUDA**
- Por que essas três?
 - ▶ Utilizaremos MPI para programação de arquiteturas com **memória distribuída (*distributed memory*)**
 - ▶ Utilizaremos Pthreads e OpenMP para programação de arquiteturas com **memória compartilhada (*shared memory*)**
 - OpenMP é de nível mais alto enquanto Pthread te dá controle fino de tudo o que está ocorrendo.
 - ▶ Utilizaremos CUDA para programação de **GPUs**

Programação/Computação Concorrente, Paralela e Distribuída são todas a mesma coisa?

- Na **computação concorrente** diversas tarefas podem estar em progresso no mesmo momento
- Na **computação paralela** um programa é composto de diversas tarefas **cooperam entre si** para resolver um problema
- Na **computação distribuída** um programa precisa colaborar com outros programas para resolver um problema

Neste curso veremos um pouco de cada mas daremos especial atenção à programação paralela e concorrente.

Atenção:

As definições acima estão longe de ser unanimidade entre os especialistas!

- Se você acha que vai conseguir escrever na gambiarra um programa paralelo eficiente pode tirar o cavalinho da chuva
- Se para programas sequenciais programar seguindo boas práticas é importante, para programas paralelos é essencial!
- Nosso trabalho vai ser casar o software com o hardware e para isto precisaremos de um elevado conhecimento de ambos.
- Utilizaremos ferramentas avançadas de programação. Para isto é esperado que você saiba:
 - ▶ C (avançado)
 - ▶ Linux (intermediário)