INTRODUÇÃO

MCZA020-13 – PROGRAMAÇÃO PARALELA

Emilio Francesquini
e.francesquini@ufabc.edu.br

17 de setembro de 2018

Centro de Matemática, Computação e Cognição
Universidade Federal do ABC

UFABC

DISCLAIMER

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

INFORMAÇÕES DE CONTATO

- Prof. Dr. Emilio Francesquini
- e.francesquini@ufabc.edu.br
- http://professor.ufabc.edu.br/~e.francesquini
- Santo André, Bloco A, Sala 531-2

Todas as informações relativas à disciplina tais como:

- Datas importantes
- Critérios de avaliação
- Bibliografia
- Avisos
- ...

Estarão disponíveis em:

http://professor.ufabc.edu.br/~e.francesquini/pp/

Ou simplesmente busque pelo meu nome e ache o link na minha página.

Multitasking

Attention, multitaskers (if you can pay attention, that is): Your brain may be in trouble.

People who are regularly bombarded with several streams of electronic information do not pay attention, control their memory or switch from one job to another as well as those who prefer to complete one task at a time, a group of Stanford researchers has found.

(...)

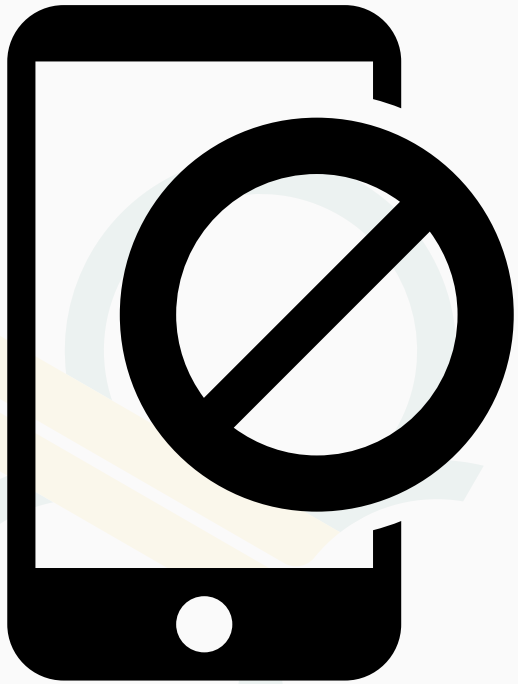
So maybe it's time to stop e-mailing if you're following the game on TV, and rethink singing along with the radio if you're reading the latest news online. By doing less, you might accomplish more.

http://news.stanford.edu/2009/08/24/multitask-research-study-082409/

O PERIGO DE FAZER VÁRIAS COISAS AO MESMO TEMPO

- Veja o vídeo de Clifford Nass (Stanford) em `https://youtu.be/PriSFBu5CLs`
- Se render às distrações do mundo digital (e-mail, mensagens instantâneas, Facebook, etc.) faz o cérebro lançar pequenas doses de dopamina
- Com o tempo, ficamos viciados nisso
- Resultado: *multitaskers* gastam muito mais poder de processamento cerebral do que *monotaskers* quando são destraídos
- Efeitos a longo prazo são difíceis de reverter

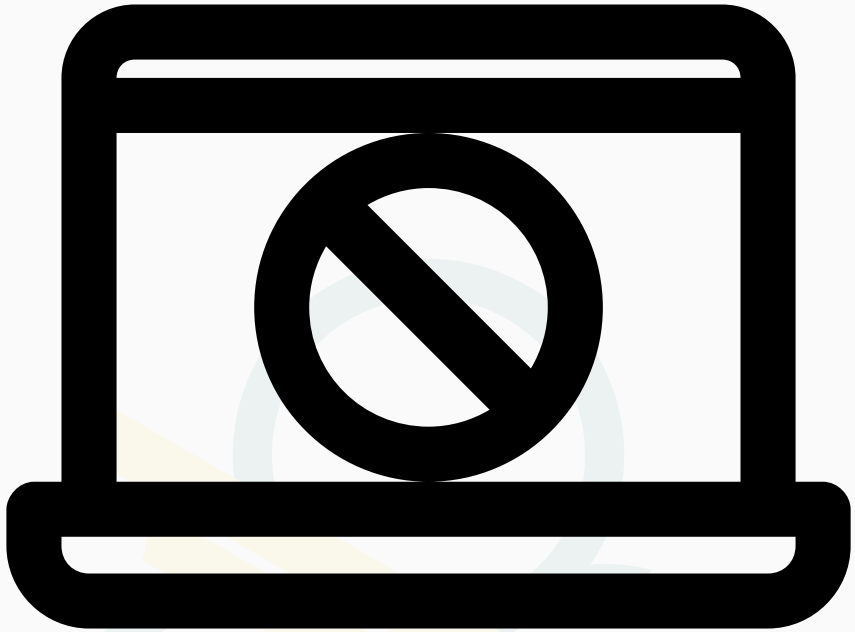
POR ISSO, NA SALA DE AULA:



No Phone by Rflor from the Noun Project

POR ISSO, NA SALA DE AULA:

POR ISSO, NA SALA DE AULA:



blocked laptop by unlimicon from the Noun Project

Contudo, em algumas aulas haverá demonstrações de uso de tecnologias de programação. Nestes momentos aqueles que quiserem seguir em seus notebooks estão liberados.

ALUNOS COM DEFICIÊNCIA

Avise seu professor **o quanto antes** sobre a necessidade de cuidados extras para acessibilidade nos casos de deficiência:

- visual,
- física,
- auditiva,
- dislexia,
- etc.

`http://proap.ufabc.edu.br/`

AULAS TEÓRICAS

- Professor: Emilio Francesquini — `e.francesquini@ufabc.edu.br`
- Aulas:
  - Segundas das 08:00 às 10:00, semanal, Sala S-301-3
  - Quintas das 10:00 às 12:00, semanal, Sala S-301-3
- Fórum, dúvidas e anúncios: Procure no TIDIA por ”2018.Q3.Paralela-Emilio”.

ATENDIMENTO

- Presencial
  - **Horários de atendimento**
    - Segunda-feira, das 10:00 às 12:00, Sala 531-2.
    - Quarta-feira, das 13:00 às 15:00, Sala 531-2.
  - **Agendado por e-mail**
  - **Em sala de aula** - Após as aulas
- Online
  - **Por e-mail**.
  - **Pelo fórum da disciplina (TIDIA)**.

HONESTIDADE ACADÊMICA

Qualquer tentativa de fraude nas provas, listas de exercícios ou projetos implicará:

- **Conceito final CF = F (reprovado)** para **TODOS** os envolvidos.
- **Possível denúncia apresentada à Comissão de Transgressões Disciplinares Discentes da Graduação**, a qual decidirá sobre a punição adequada à violação que pode resultar em **advertência, suspensão ou desligamento**, de acordo com os artigos 78-82 do Regimento Geral da UFABC.
- **Possível denúncia apresentada à Comissão de Ética da UFABC**, de acordo com o artigo 25 do Código de Ética da UFABC.

HONESTIDADE ACADÊMICA

- Diversos professores se reuniram e escreveram um Código de Honra
  - `http://professor.ufabc.edu.br/~e.francesquini/codigodehonra/`
- Nesta disciplina seguiremos este código
- Leiam o texto completo e, em caso de dúvidas, perguntem ao professor

Regra 1: Você não pode enviar para avaliação um trabalho que não seja de sua própria autoria ou que seja derivado/baseado em soluções elaboradas por outros.

Regra 2: Você não pode compartilhar a sua solução com outros alunos nem pedir aos seus colegas que compartilhem as soluções deles com você.

Regra 3: Nos trabalhos enviados para avaliação você deve indicar eventuais assistências que você tenha recebido.

CRITÉRIOS DE AVALIAÇÃO

A avaliação da disciplina será composta por duas notas, uma referente à teoria e outra à prática. Considere:

- N_F é a nota final;
- N_{Pv} é a nota das provas;
- N_{Pj} é a nota dos projetos.

A nota final N_F será determinada da seguinte maneira:

$$N_F = \begin{cases} \mathbf{min}\{N_{Pv}, N_{Pj}\}, & \text{se } N_{Pv} < 5 \text{ ou } N_{Pj} < 5 \\ 0.6 \cdot N_{Pv} + 0.4 \cdot N_{Pj}, & \text{caso contrário} \end{cases}$$

O conceito final C_F será obtido de acordo com a equação abaixo:

$$C_F = \begin{cases} \mathbf{O}, & \text{se ausência total exceder 25\%} \\ \mathbf{F}, & \text{se } N_F \in [0,0; 5,0) \\ \mathbf{D}, & \text{se } N_F \in [5,0; 6,0) \\ \mathbf{C}, & \text{se } N_F \in [6,0; 7,0) \\ \mathbf{B}, & \text{se } N_F \in [7,0; 8,5) \\ \mathbf{A}, & \text{se } N_F \in [8,5; 10,0] \end{cases}$$

A nota das provas N_{PV} será formada por duas provas P_1 e P_2. Todas as provas serão efetuadas em sala de aula, sem qualquer tipo de consulta.

Haverá também uma prova subsitutiva P_S que será aberta a todos os interessados, ainda que eles tenham feito tanto a P_1 quanto a P_2.

ATENÇÃO — A nota da P_S será utilizada obrigatoriamente em substituição à menor nota entre P_1 e P_2 ainda que isto diminua a nota final do aluno!

$$N_{PV} = \begin{cases} \frac{2 \cdot P_S + 3 \cdot P_2}{5}, & \text{caso tenha feito a } P_S \text{ e } P_1 < P_2 \\ \frac{2 \cdot P_1 + 3 \cdot P_S}{5}, & \text{caso tenha feito a } P_S \text{ e } P_2 \leq P_1 \\ \frac{2 \cdot P_1 + 3 \cdot P_2}{5}, & \text{caso contrário} \end{cases}$$

Teremos dois projetos durante o quadrimestre (Pr_1 e Pr_2) de igual peso. Sua nota será, então, calculada pela seguinte equação:

$$N_{Pj} = \frac{Pr_1 + Pr_2}{2}$$

RECUPERAÇÃO

A Resolução ConsEPE nº 182 garante a todos os alunos com C_F igual a **D** ou **F** o direito a fazer uso de mecanismos de recuperação.

A recuperação será feita através de uma prova P_R, sem consulta, e a sua nota será utilizada para compor a o conceito pós-recuperação C_R conforme as equações abaixo:

$$N_R = \frac{P_R + N_F}{2}$$

Caso 1 $C_F = D$**:**

$$C_R = \begin{cases} \textbf{C}, & \text{se } N_R \geq 6,0 \\ \textbf{D}, & \text{caso contrário} \end{cases}$$

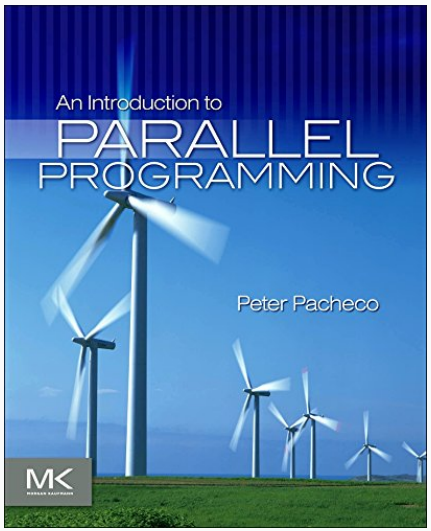
Caso 2 $C_F = F$**:**

$$C_R = \begin{cases} \textbf{D}, & \text{se } N_R \geq 5,0 \\ \textbf{F}, & \text{caso contrário} \end{cases}$$

DATAS IMPORTANTES

- **Prova 1** — 25/10/2018
- **Prova 2** — 06/12/2018
- **Prova Substitutiva** — 13/12/2018
- **Prova de Recuperação** — 18/12/2018
- **Projeto 1** — 28/10/2018
- **Projeto 2** — 09/12/2018

LIVRO TEXTO



- Peter Pacheco. **An Introduction to Parallel Programming**. Second Edition.
- `http://biblioteca.ufabc.edu.br/index.php?codigo_sophia=13315`
- Tópicos:
  - Introdução;
  - Arquiteturas de computadores;
  - Paralelismo em Arquiteturas de Memória Distribuída;
  - Paralelismo em Arquiteturas de Memória Compartilhada;

BIBLIOGRAFIA ADICIONAL

- Thomas Rauber, Gudula Rünger. **Parallel Programming: For Multicore and Cluster Systems**. Second Edition
  - Link Biblioteca: `http://biblioteca.ufabc.edu.br/index.php?codigo_sophia=87487`
  - O PDF do livro **pode ser baixado diretamente (gratuitamente)** daqui: `http://dx.doi.org/10.1007/978-3-642-37801-0`
    - **Atenção!** Para baixar gratuitamente você deve fazer o download **a partir da rede UFABC**
- Ananth Grama, Anshul Gupta, George Karypis, Vipin Kumar. **Introduction to parallel computing**. Second Edition.
  - `http://biblioteca.ufabc.edu.br/index.php?codigo_sophia=6678`

Programação Paralela

QUAL A NECESSIDADE DE COMPUTAÇÃO 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?

QUAL A NECESSIDADE DE COMPUTAÇÃO 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?
  - **60x** em 10 anos *vs.* **6x**
- Limites do desenvolvimento levaram aos multi-core

TRÊS PERGUNTAS IMPORTANTES

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%?

Pergunta 1: e daí que são apenas 20%?

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

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

EXEMPLO: PROBLEMAS EM ESCALA DA WEB

- Características
  - Definitivamente *data-intensive*
  - Mas podem também ser *processing-intensive*
- Exemplos:
  - *Crawling*, indexação, busca, mineração de dados da web
  - Pesquisa em biologia computacional na era “pós-genômica”
  - Processamento de dados científicos (física, astronomia, etc.)
  - Redes de sensores
  - Aplicações Web 2.0
  - etc.

DE QUAL VOLUME DE DADOS ESTAMOS FALANDO?

Problemas da ordem de petabytes!

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

DE QUAL VOLUME DE DADOS ESTAMOS FALANDO?

Muitos, mas muitos dados

- O Google processa cerca de **20 petabytes** de dados por dia (2008)
- O Wayback Machine tem cerca de **3 petabytes + 100 terabytes/dia** (mar/2009)
- O Facebook tem cerca de **2,5 petabytes de dados de usuários + 15 terabytes/dia** (abr/2009)
- 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?

Muitos, mas muitos dados

- O Google processa cerca de **20 petabytes** de dados por dia (2008)
- O Wayback Machine tem cerca de **3 petabytes + 100 terabytes/dia** (mar/2009)
- O Facebook tem cerca de **2,5 petabytes de dados de usuários + 15 terabytes/dia** (abr/2009)
- 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?

Muitos, mas muitos dados

- O Google processa cerca de **20 petabytes** de dados por dia (2008)
- O Wayback Machine tem cerca de **3 petabytes + 100 terabytes/dia** (mar/2009)
- O Facebook tem cerca de **2,5 petabytes de dados de usuários + 15 terabytes/dia** (abr/2009)
- 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?

Muitos, mas muitos dados

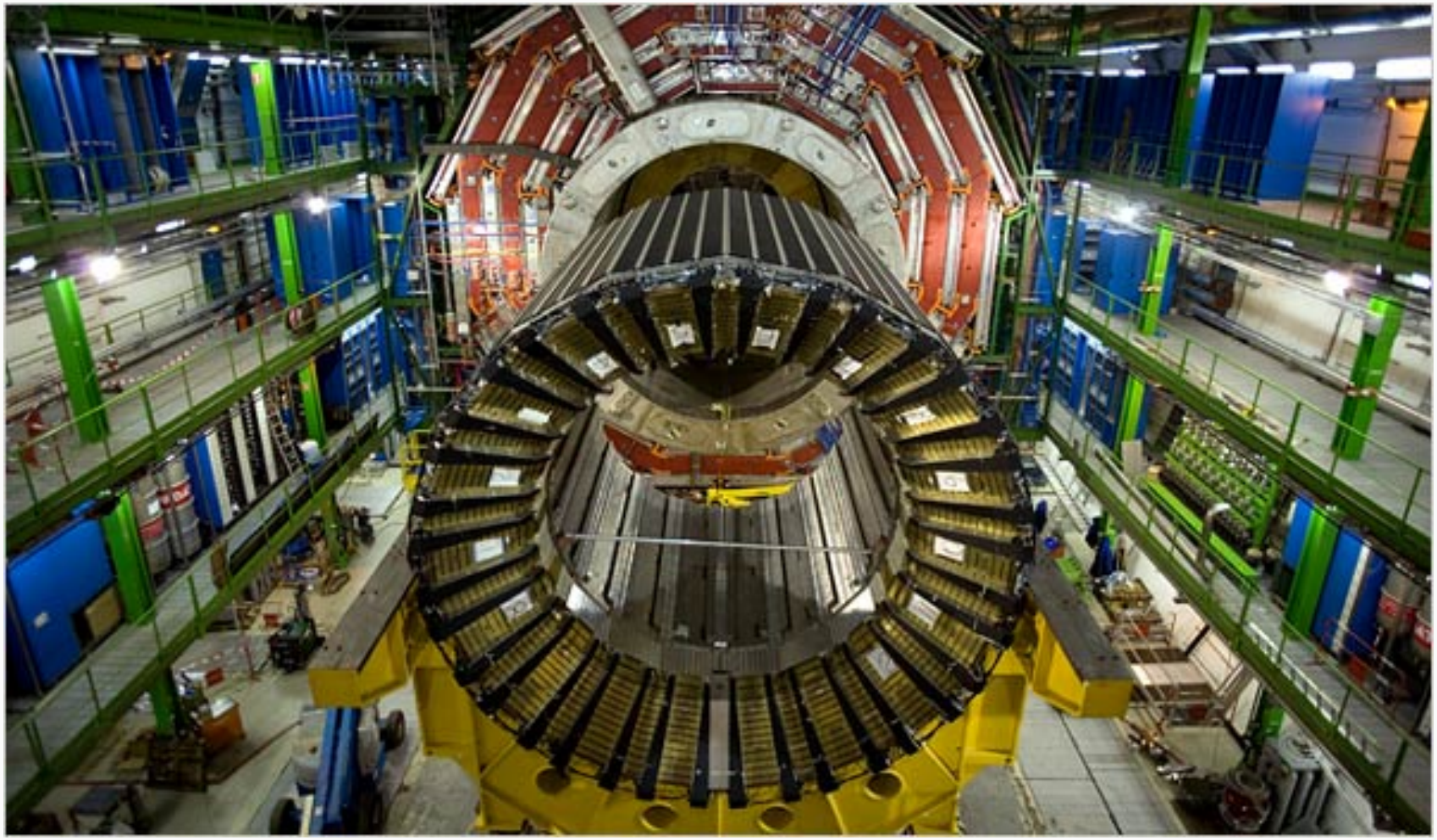
- O Google processa cerca de **20 petabytes** de dados por dia (2008)
- O Wayback Machine tem cerca de **3 petabytes + 100 terabytes/dia** (mar/2009)
- O Facebook tem cerca de **2,5 petabytes de dados de usuários + 15 terabytes/dia** (abr/2009)
- O site eBay tem cerca de **6,5 petabytes de dados dos 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?

Muitos, mas muitos dados

- O Google processa cerca de **20 petabytes** de dados por dia (2008)
- O Wayback Machine tem cerca de **3 petabytes + 100 terabytes/dia** (mar/2009)
- O Facebook tem cerca de **2,5 petabytes de dados de usuários + 15 terabytes/dia** (abr/2009)
- 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?



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

O QUE SE FAZ COM TANTOS DADOS?

- 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

O QUE SE FAZ COM TANTOS DADOS?

- 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

O QUE SE FAZ COM TANTOS DADOS?

- 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

O QUE SE FAZ COM TANTOS DADOS?

- 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

O QUE SE FAZ COM TANTOS DADOS?

- 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?

GRANDES DATA CENTERS

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

GRANDES DATA CENTERS

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

GRANDES DATA CENTERS

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

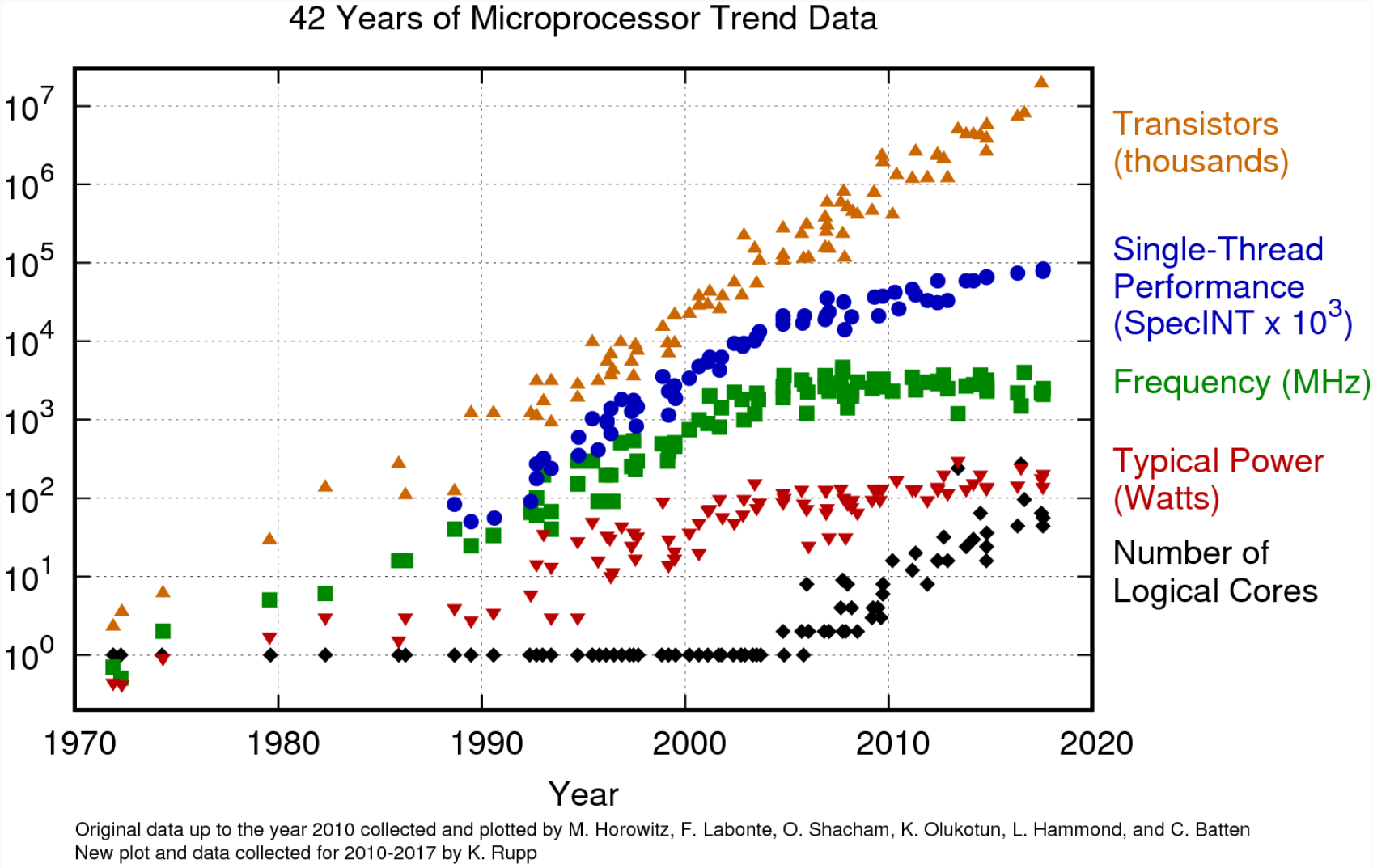
GRANDES DATA CENTERS

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

Só o Google tem treze desses espalhados pelo mundo!

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?

EVOLUÇÃO DOS PROCESSADORES



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

FREE LUNCH IS OVER...

- 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?

MÚLTIPLOS CORES NÃO RESOLVEM TUDO...

- 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`)

EXEMPLO: SOMA DE UMA LISTA DE NÚMEROS

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

SOMA PARALELA - VERSÃO 1

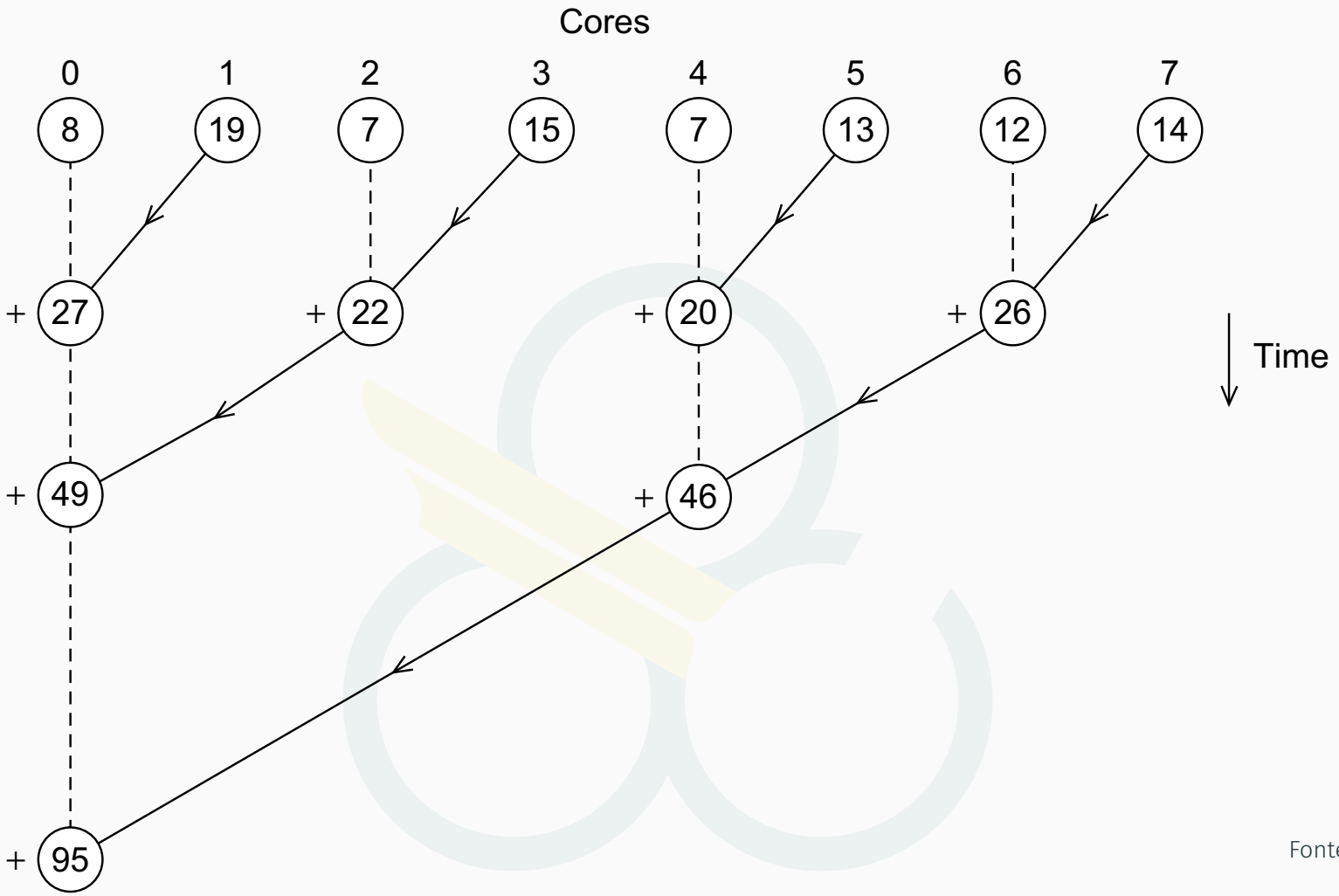
```
if (I 'm the master core) {
    sum = my x;
    for each core other than myself {
        receive value from core;
        sum += value;
    }
} else {
    send my x to the master;
}
```

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?

Pergunta 2:
Quantas comunicações cada um dos processos faz no caso geral? É possível melhorar?

SOMA PARALELA - VERSÃO 2



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.

PROBLEMAS VERGONHOSAMENTE PARALELOS

- Alguns problemas são classificados como vergonhosamente paralelos (*embarassingly parallel*)
  - Você também vai ver por ai “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!

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

Nosso exemplo da soma emprega diversos mecanismos de coordenação:

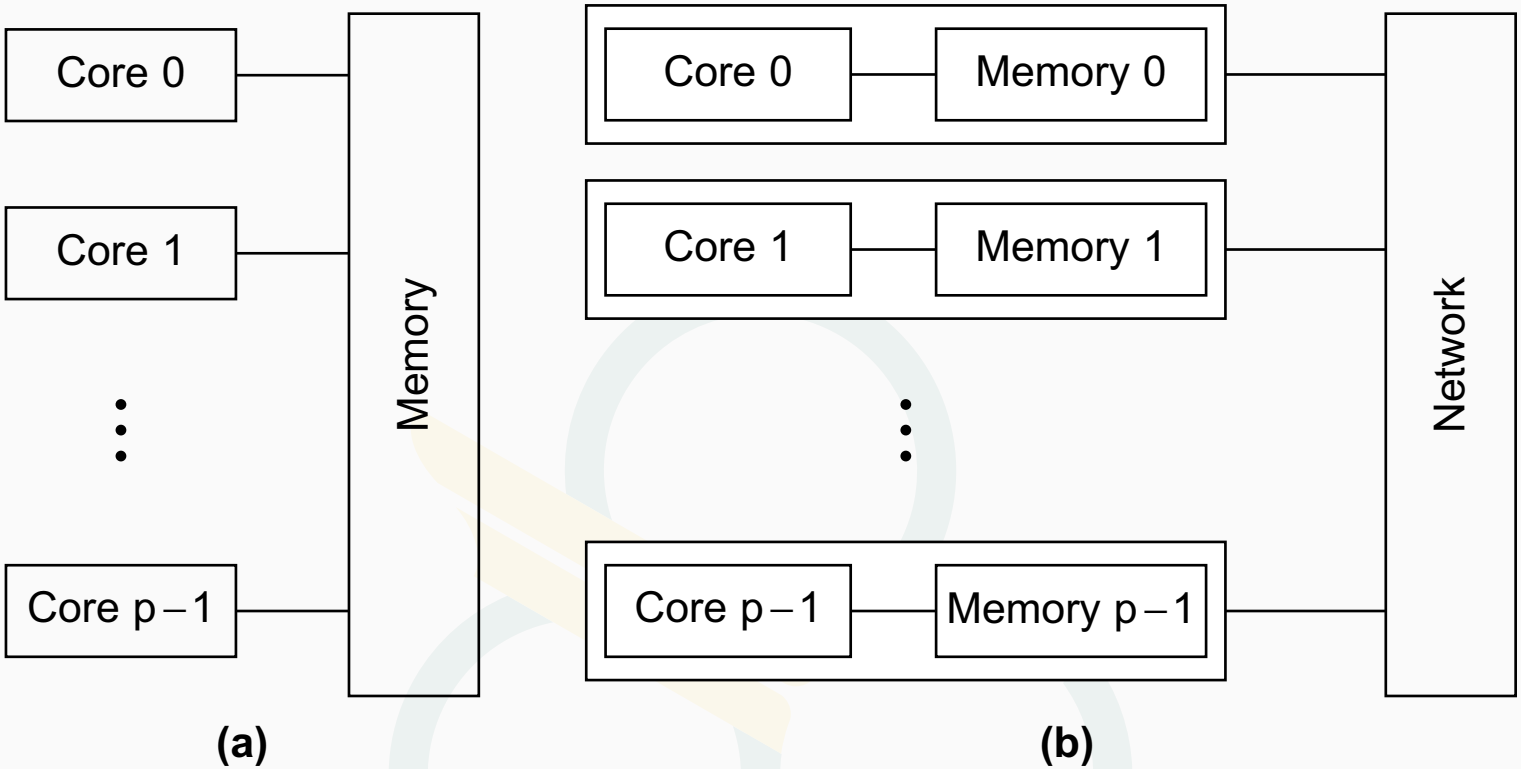
- Comunicação – um ou mais cores enviam suas somas parciais para outros cores
- Balanceamento de carga (*load balancing*) – Apesar de termos fórmulas fechadas para o trabalho a ser desempenhado por cada core, queremos que o trabalho seja bem distribuído.
  - Minimização do caminho crítico (*critical 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 Dezembro...

O QUE VEREMOS NESTE CURSO?

- 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**. Se você:
  - Nunca programou em C ou;
  - Não fez *Programação Estruturada* ou;
  - Tem medo de ponteiros ou;
  - Não entende a diferença entre:
    - Os `*` de: `void* x = v[*p + 1 * 3];`
    - Os `&` de: `int x = (&a & 0xFFFF) && b;`
  - **Você terá MUITA dificuldade neste curso!**
  - **Comece a estudar agora!**

COM QUAIS TIPOS DE MÁQUINAS TRABALHAREMOS?



(a) memória compartilhada (b) memória distribuída Fonte: [PP]

Pergunta:
Em qual delas roda o seu APP Angry Birds? Em qual delas roda sua busca na Web?

QUAIS FERRAMENTAS UTILIZAREMOS?

- Aprenderemos as seguintes ferramentas
  - Message Passing Interface - MPI
  - POSIX Threads - Pthreads
  - OpenMP
- 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.

MAS ESPERA AÍ!!!

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!

RESUMO DA ÓPERA

- 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)