

# SuperComputação

## Aula 5 – Modelo fork-join

2020 – Engenharia

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

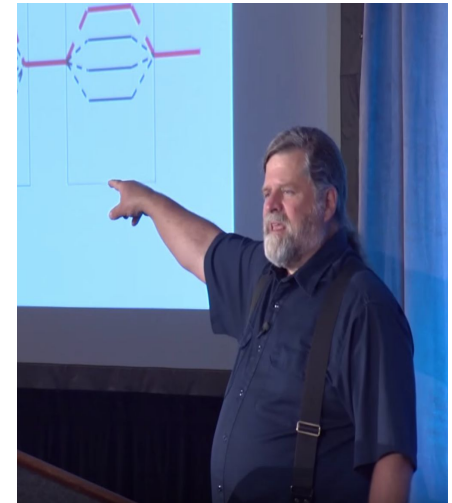
# Fontes importantes

## A brief Introduction to parallel programming

**Tim Mattson**

**Intel Corp.**

**timothy.g.mattson@intel.com**



### Vídeos:

<https://www.youtube.com/watch?v=pRtTIW9-Nr0>

<https://www.youtube.com/watch?v=LRsQHDAqPHA>

<https://www.youtube.com/watch?v=dK4PITrQtjY>

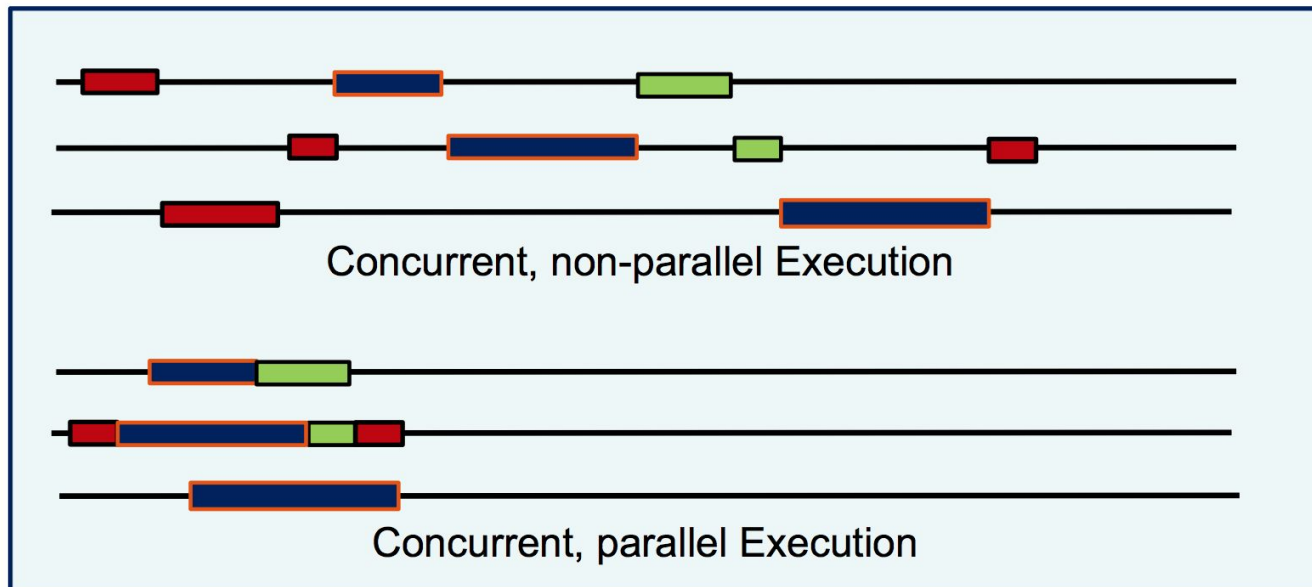
[https://www.youtube.com/watch?v=WvoMpG\\_QvBU](https://www.youtube.com/watch?v=WvoMpG_QvBU)

### Slides:

[http://extremecomputingtraining.anl.gov/files/2016/08/Mattson\\_830aug3\\_HandsOnIntro.pdf](http://extremecomputingtraining.anl.gov/files/2016/08/Mattson_830aug3_HandsOnIntro.pdf)

# Paralelismo e concorrência

- **Concorrência:** condição de um sistema na qual múltiplas tarefas estão logicamente ativas ao mesmo tempo.
- **Paralelismo:** condição de um sistema na qual as múltiplas tarefas estão realmente ativas ao mesmo tempo.



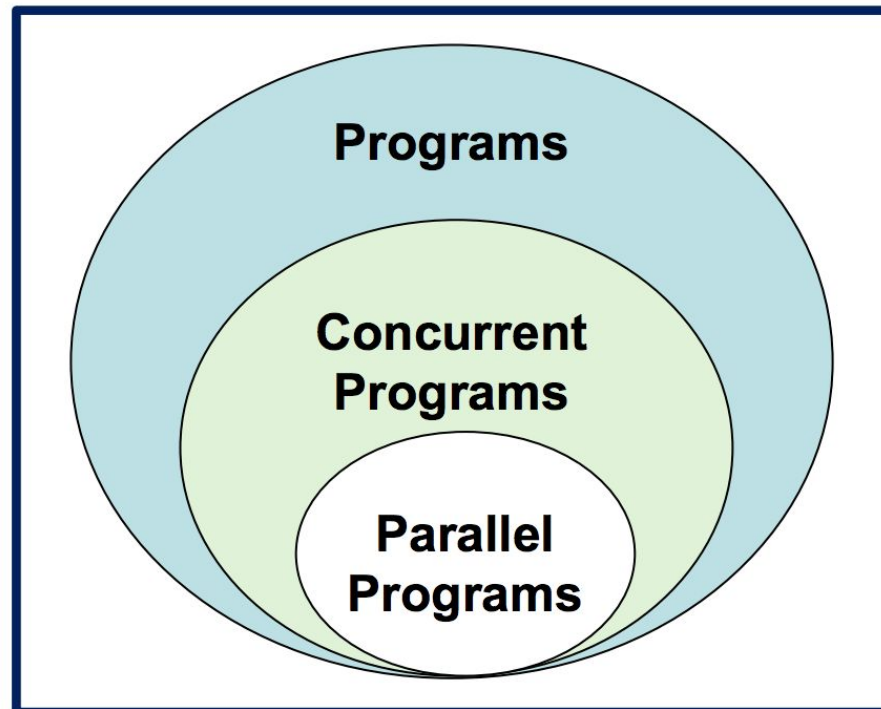
# Concorrência sem paralelismo

- Sistemas orientados a eventos:
  - Sistemas embarcados: sensores e atuadores
  - Comunicação via rede (web servers, crawlers, etc)
  - Banco de dados
  - Produtor consumidor

Objetivo: fazer com que várias tarefas avancem de maneira conjunta e que nenhuma delas fique sem rodar.

# Paralelismo e concorrência

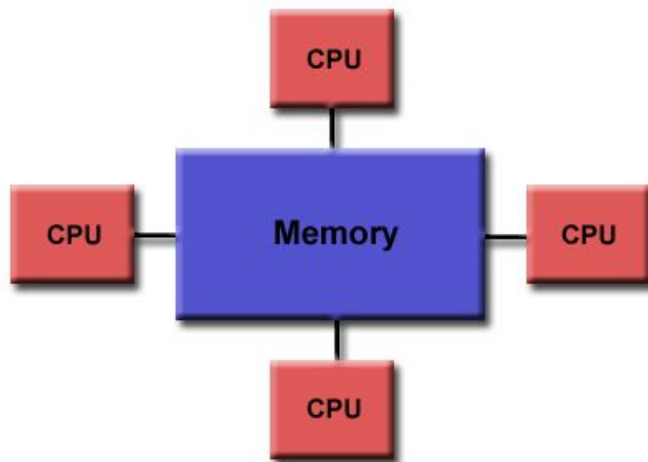
- **Paralelismo:** usado para
  - fazer mais trabalho em menos tempo
  - tratar problemas grandes



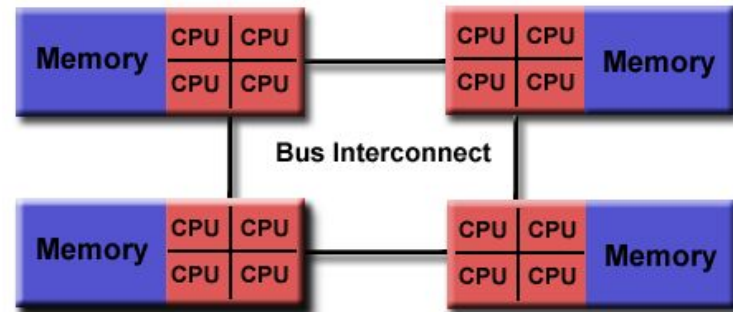
# Computação paralela atual

- Multicore com OpenMP
- Cluster com MPI
- GPGPU com CUDA ou **OpenCL**
- FPGA (Google TPU) com **OpenCL** (Embarcados Avançados)

# Sistemas Multi-core

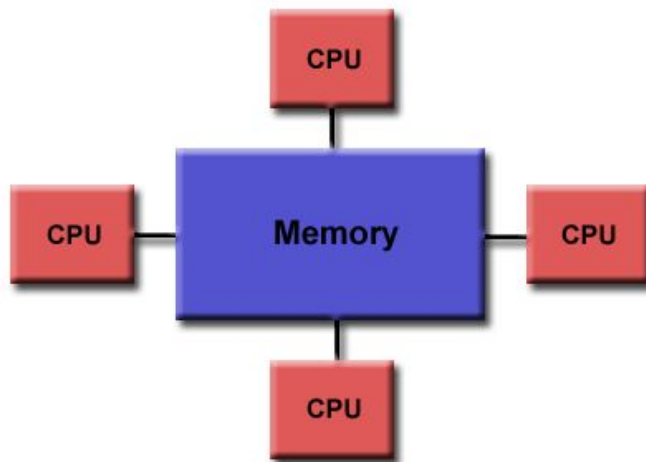


**Uniform Memory Access**

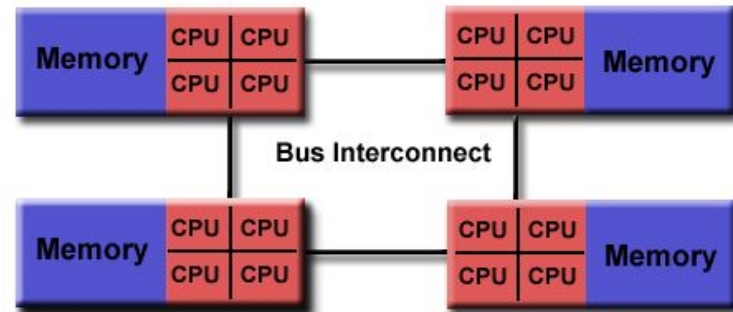


**Non-Uniform Memory Access**

# Sistemas Multi-core



**Uniform Memory Access**



**Non-Uniform Memory Access**



# Sistemas Multi-core (Tarefas)

- Processos
- Threads

# Processos

Principal mecanismo de execução de tarefas

Tarefas completamente isoladas

É possível a comunicação entre processos via

- Entrada e saída padrão
- Passagem de mensagens
- Memória (**explicitamente**) compartilhada

# Processos

Criação (POSIX):

*pid\_t fork();*

Cria um novo processo duplicando todos os dados do processo chamador (pai). Retorna duas vezes.

1. No pai: pid do filho
2. No filho: 0

# Threads

Cada processo tem um ou mais threads

Compartilham memória

Problema com sincronização ao acesso a objetos compartilhados

# Threads

Em C: POSIX Threads (obsoleto?)

Em C++11: cabeçalho <thread>

1. `std::thread` contém implementação simples de threads
2. `std::this_thread` é usado para referenciar o thread atual em uma função

# Threads vs Processos

## Threads:

- Ambientes multi-core
- Modelo fork-join

## Processos:

- Pode ser usado em ambientes distribuídos
- Compartilhamento explícito de dados

# Threads vs Processos

## Threads:

- Ambientes multi-core
- Modelo fork-join

## Processos:

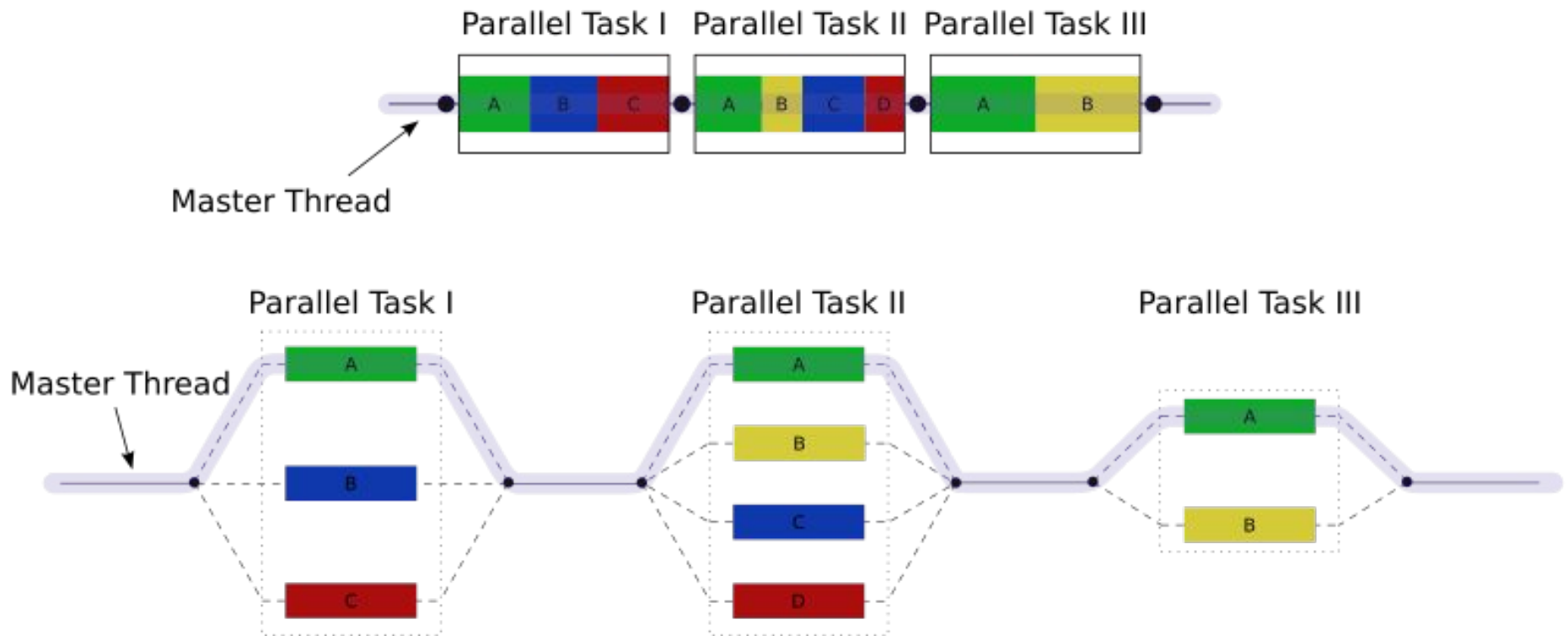
- Pode ser usado em ambientes distribuídos
- Compartilhamento explícito de dados

# Sistemas Multi-core (Tarefas)

1. Memória Compartilhada
2. Divisão de tarefas
3. O quê pode ser paralelizado?



# Modelo fork-join



# Modelo fork-join

1. Divide tarefa em pedaços
2. Resolve cada uma de maneira independente
3. Junta os resultados

# Cálculo do pi

## Recorde atual:

por Peter Trueb (Dectris)

22,459,157,718,361 dígitos

105 dias de processamento

### Servidor:

Dell PowerEdge R930

4 hyper-threaded 18-core Intel Xeon E7-8890 v3 @ 2.5 GHz

1.25 TB RAM

### Armazenamento

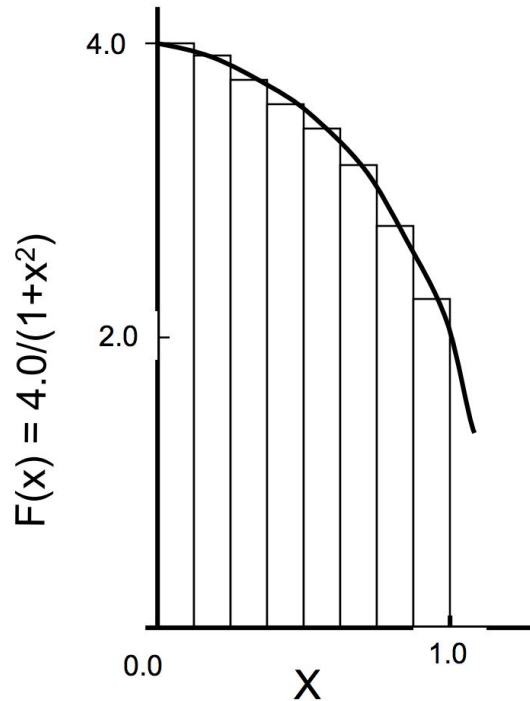
20 x Seagate Enterprise NAS 6 TB

4 GB/s bandwidth

60 TB Backup Storage

Código usado:  $\gamma$ -cruncher por Alexander J. Yee baseado no algoritmo de Chudnovsky.

# Cálculo do pi



Matematicamente

$$\int_0^1 \frac{4.0}{(1+x^2)} dx = \pi$$

A integral pode ser aproximada por uma soma de retângulos

$$\sum_{i=0}^N F(x_i) \Delta x \approx \pi$$

Cada retângulo tem largura  $\Delta x$  e altura  $F(x_i)$  no meio do intervalo  $i$

# Cálculo do pi (Orfali)

$$\left(\frac{\sin x}{\cos x}\right)' = \sec^2 x$$

$$\frac{\pi}{4} = \int_0^1 \frac{1}{1+x^2} dx = \arctg x \Big|_0^1$$

$$= \arctg 1 - \arctg 0$$

$$\frac{\pi}{4} - 0$$

$$\alpha = \arctg x$$

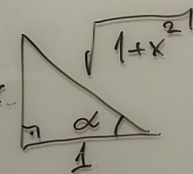
$$\cos \alpha = \frac{1}{\sqrt{1+x^2}}$$

$$\sec \alpha = \sqrt{1+x^2}$$

$$\sec^2 \alpha = 1+x^2$$

$$\frac{d}{dx} (\arctg x) = \frac{1}{1+x^2}$$

$$\frac{d}{dx} \arctg x = \frac{1}{1+x^2}$$

$$\frac{d}{dx} \arctg x = \frac{1}{1+x^2}$$


# Modelo fork-join raiz

Atividade de hoje: implementar modelo fork-join na mão usando `std::thread`.

# Referências

- Livros:
  - Hager, G. ; Wellein, G. **Introduction to High Performance Computing for Scientists and Engineers**. 1ª Ed. CRC Press, 2010.
- Internet:
  - [https://en.wikipedia.org/wiki/Fork%E2%80%93join\\_model](https://en.wikipedia.org/wiki/Fork%E2%80%93join_model)
  - <https://en.cppreference.com/w/cpp/thread/thread>
  - <https://en.cppreference.com/w/cpp/thread>

# Inspire

[www.inspire.edubr](http://www.inspire.edubr)