Programação Paralela e Distribuída

Prof. Hugo Alberto Perlin

Programação Paralela

- Dividir uma tarefa computacional em *n* partes independentes (sendo que *n* >= 2), de forma que as partes possam ser executadas em paralelo (ao mesmo tempo em núcleos de processamento diferentes)

Programação Paralela

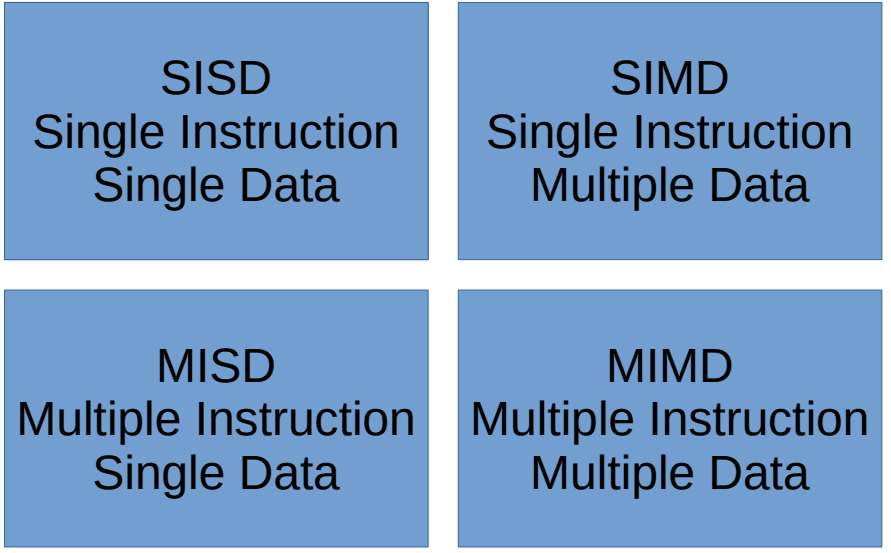
- “Se um único computador (processador) consegue resolver um problema em N segundos, podem N computadores (processadores) resolver o mesmo problema em 1 segundo?”

Programação Paralela

- Sistemas paralelos possuem uma dualidade de caracterização:
  - Baseada na arquitetura do sistema de computação
  - Baseada nos paradigmas de programação paralela

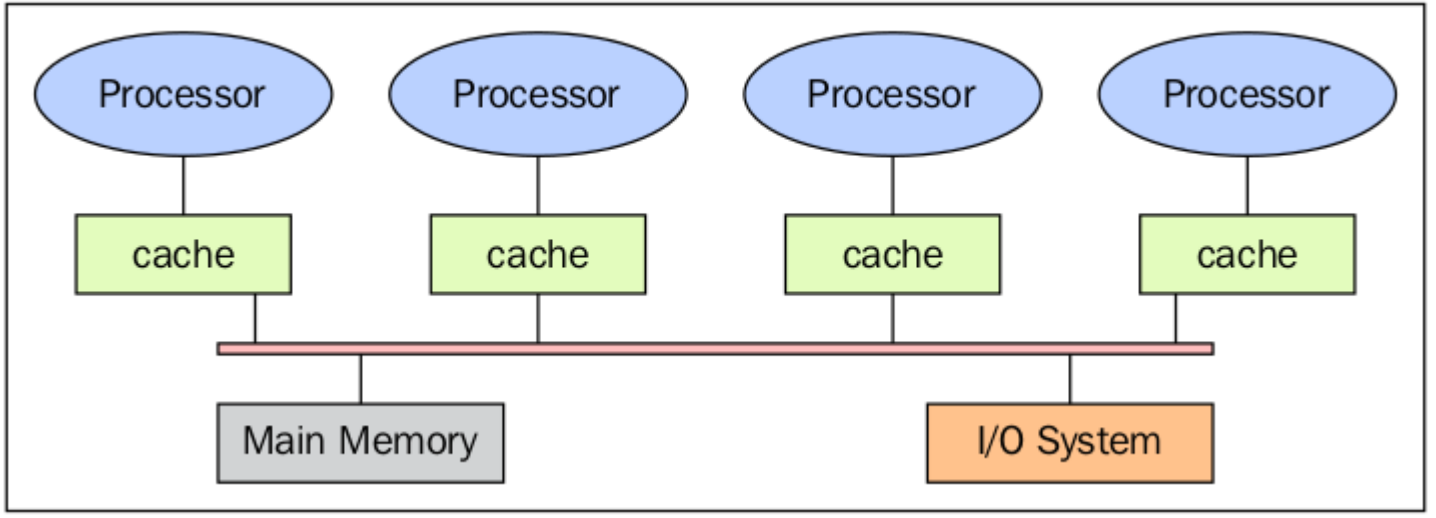
Programação Paralela – Arquitetura dos Sistemas

- Arquitetura de memória para computação paralela
- Taxonomia de Flynn



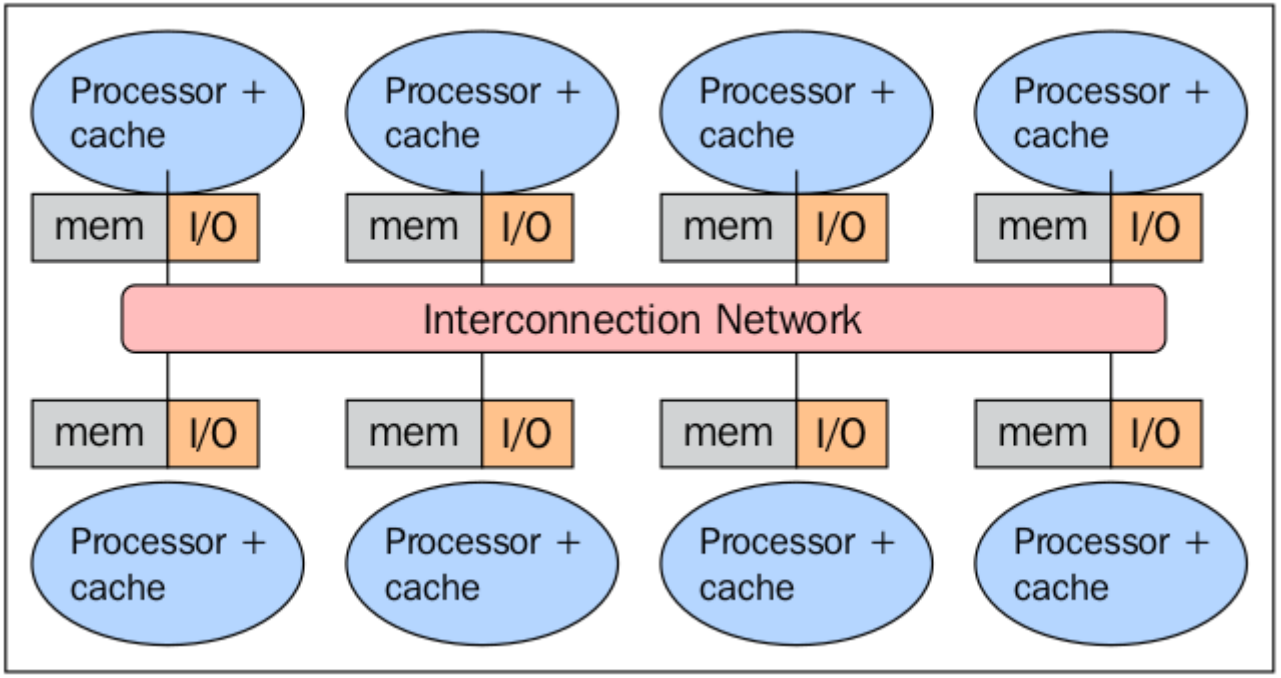
Programação Paralela – Arquitetura dos Sistemas

- Memória Compartilhada



Programação Paralela – Arquitetura dos Sistemas

- Memória Distribuída



Programação Paralela – Paradigmas de Programação

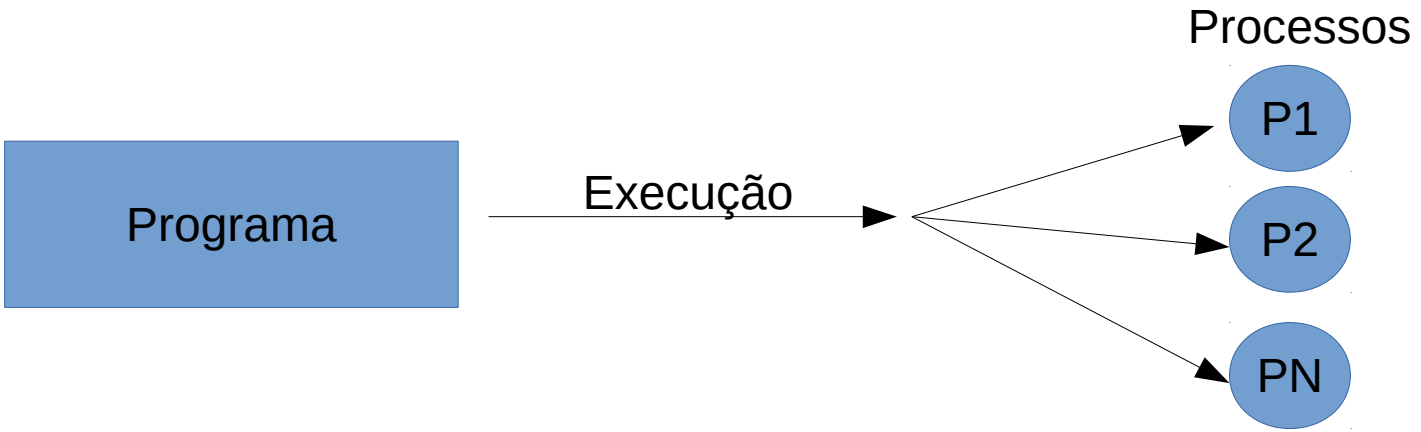
- Modelo Memória Compartilhada: neste modelo as tarefas compartilham uma única área de memória compartilhada, onde o acesso (escrita e leitura) aos recursos compartilhados é assíncrono.
- Modelo Multi-thread: um processo pode possuir vários fluxos de execução, geralmente este modelo é utilizado em arquiteturas de memória compartilhada.
- Modelo de Troca de Mensagens: este modelo é utilizado no caso em que cada processador possui sua própria memória (sistemas distribuídos). Um protocolo de troca de mensagem deve ser utilizado para trafegar dados.
- Modelo de Paralelização de Dados: neste modelo diversas tarefas operam sobre uma mesma estrutura de dados porém em uma porção diferente. As GPUs utilizam este modelo de computação.

Modelo Multi-thread – Programa vs Processo

- Programa
  - conjunto de instruções numa linguagem de alto nível ou de máquina
- Processo
  - Execução do programa
  - Cada processo tem seu conjunto de instruções, seu contador de programa e sua área de memória

Modelo Multi-thread – Programa vs Processo

- Todo processo possui um identificador único (PID)
- Um programa executado N vezes, possui N processos em execução



Modelo Multi-thread – Troca de Contexto

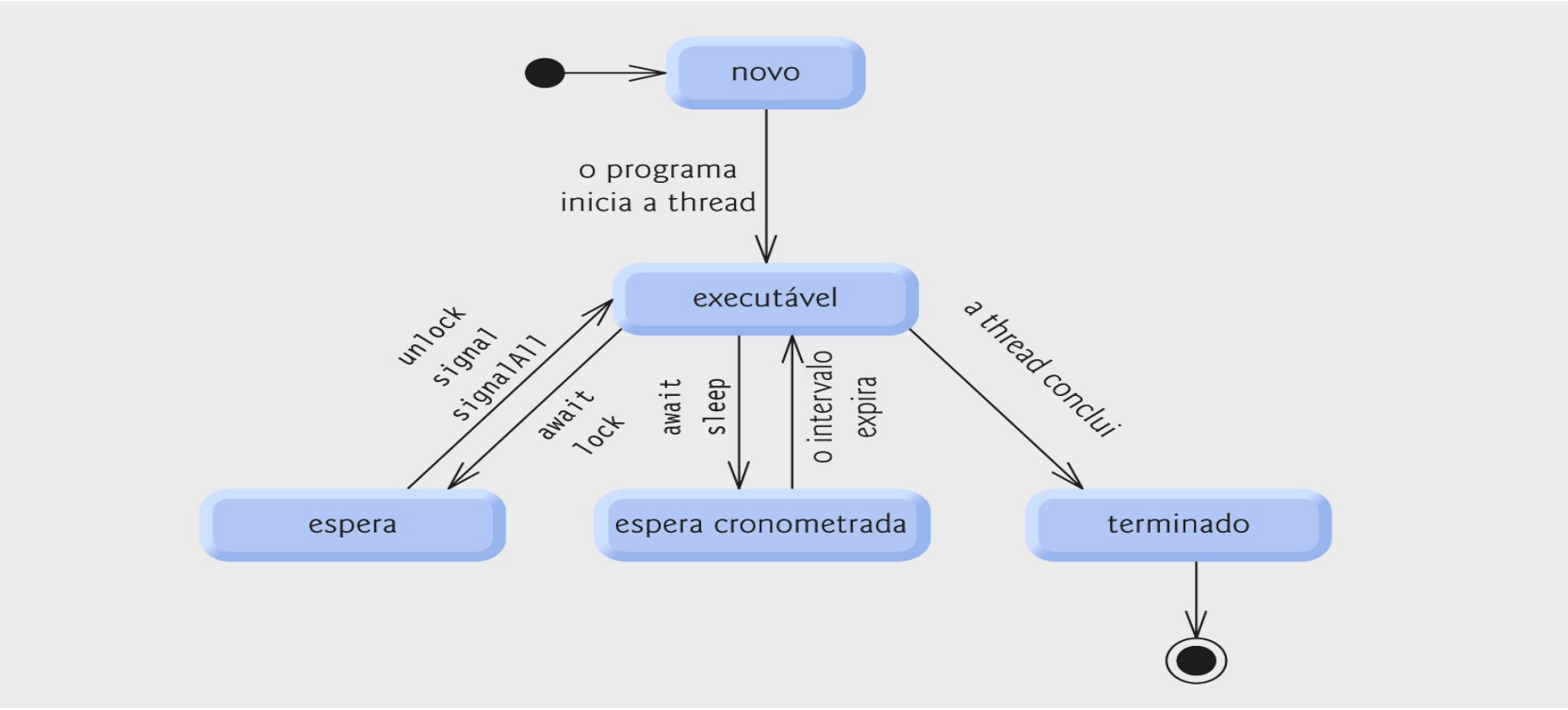
- Nos sistemas modernos, N processos competem pelo uso da CPU
- Cada processo é responsável por manter seu contador de programa e sua memória
- Um processo pode ser interrompido pelo SO, e ser retirado do processador, tendo que aguardar momento oportuno para ser executado novamente
- Este processo é chamado de troca de contexto

Modelo Multi-thread – Threads

- Todo processo possui ao menos um fluxo de execução (thread)
- Nos SOs modernos é possível criar múltiplas threads
- Cada thread é parte de um processo
  - Compartilham recursos: memória e arquivos abertos
- Cada thread tem seu estado e segue um ciclo de vida
- Por que utilizar múltiplas threads?

Thread

- Ciclo de vida de uma thread



Modelo Multi-thread – Threads

- Em sistemas monoprocessados, somente um processo é executado por vez e dentro do processo somente uma thread também é executada por vez
- Em sistemas multiprocessados, *n* threads serão executadas em paralelo

Modelo Multi-thread – Threads

- Threads são consideradas processos leves
- A criação e destruição é mais rápida, gerando menos overhead
- A troca de contexto entre threads é mais rápida que a troca de contexto entre processos
- Permitem um melhor uso de sistemas multiprocessados

Threads em Python

- Utilizamos o pacote *threading*
- É possível criar threads utilizando um método *threading.Thread(group=None,target=None,name=None,args=None,kwargs=None)*
- *Exemplo*

Threads em Python

- É possível criar threads extendendo a classe Thread

```
class Tarefa(Thread):
    def __init__(self):
        Thread.__init__(self)
```

Exercícios

- Crie um programa que leia um arquivo texto e uma palavra do teclado. O programa deve mostrar quais linhas a palavra aparece. Faça isso de modo que cada thread procure numa porção de até 10 linhas do arquivo.
- Faça um programa que dada uma lista de 5 inteiros grandes, verifique se cada um dos inteiros é primo. A verificação deve ser feita por threads.