Arquiteturas Paralelas e Distribuídas

Programação Paralela e Distribuída

Eduardo Furlan Miranda

- **O desafio**: aplicações modernas exigem alto desempenho, escalabilidade e confiabilidade
- **A solução**: sistemas paralelos e distribuídos, quebrando as barreiras do processamento sequencial
- **Como funciona**: execução simultânea de tarefas (paralelismo) e coordenação entre múltiplos computadores (distribuição)
- **Benefícios**: melhor tempo de resposta, maior capacidade de processamento, tolerância a falhas

Programação Paralela e Distribuída

- Programação Paralela
  - Aplicações que exigem alto desempenho em um único sistema
  - Problemas que podem ser divididos em subproblemas independentes
  - Processamento de grandes volumes de dados
- Programação Distribuída
  - Sistemas de grande escala com alta disponibilidade
  - Aplicações que exigem tolerância a falhas
  - Cálculos intensivos que podem ser divididos em partes independentes

Motivação

- Necessidade de maior poder computacional
- Limitações do processamento sequencial
- Aplicações que exigem alta performance
  - Fim da Lei de Moore
    - A busca por alternativas ao aumento da velocidade dos processadores impulsiona o paralelismo

Programação Paralela

- Aplicações são executadas paralelamente
- Em um mesmo processador
- Em uma máquina multiprocessada
- Em um grupo de máquinas interligadas que se comporta como uma só máquina

Programação Distribuída

- Aplicações são executadas em máquinas diferentes interligadas por uma rede
- Intranets
- Internet
- Outras redes públicas ou privadas

Programação Paralela e Distribuída

- A combinação de hardware paralelo (multi-core/multi-processador),
  - sistemas operacionais que gerenciam threads (e processos) eficientemente
    - e bibliotecas que abstraem a complexidade da programação paralela
      - permite a execução simultânea de partes de um mesmo programa,
        - explorando o paralelismo para aumentar o desempenho

- A escolha entre programação paralela e distribuída
  - depende das características específicas da aplicação
    - tamanho do problema
    - recursos disponíveis
    - requisitos de desempenho
- Em muitos casos, uma combinação de ambas as abordagens pode ser utilizada para obter o melhor resultado

Arquiteturas

- Memória Compartilhada (Multicore/SMP).
- Memória Distribuída (Clusters).
- Modelos híbridos.
- Cliente-Servidor.
- Peer-to-Peer (P2P).
- Computação em Nuvem.

Tipos de Arquiteturas Paralelas

- Memória Compartilhada (Multiprocessadores)
  - Múltiplos processadores acessam a mesma memória
  - Comunicação rápida e eficiente
  - Ex.: Sistemas multi-core, servidores SMP
- Memória Distribuída (Multicomputadores)
  - Cada processador tem sua própria memória
  - Comunicação por troca de mensagens
  - Ex.: Clusters, computação em nuvem

Programação Paralela e Distribuída

- Computação Paralela
  - Memória Compartilhada
    - Vários processadores acessam a mesma área de memória
  - Comunicação implícita
    - Leitura e escrita em memória compartilhada
  - Ex.: Threads dentro de um mesmo processo
- Computação Distribuída
  - Memória Distribuída
    - Cada processador possui sua própria memória local
  - Comunicação explícita por troca de mensagens
  - Ex.: Computadores conectados em uma rede

Diferenças entre Paralelo e Distribuído

- Acoplamento (forte vs. fraco)
- Comunicação (memória compartilhada vs. mensagens)
- Escopo (único sistema vs. múltiplos sistemas)

Desafios

- Gerenciamento de concorrência
  - Condições de corrida
    - múltiplos threads acessando o mesmo dado compartilhado
  - Deadlocks (impasses)
    - threads ficam bloqueados esperando uns pelos outros
- Comunicação e sincronização
- Tolerância a falhas
- Escalabilidade

Aplicações

- Simulações científicas
- Processamento de Big Data
- Computação gráfica e renderização
- Sistemas de tempo real (ex.: SCADA)
- Processamento de linguagem natural
- Meteorologia

Programação concorrente

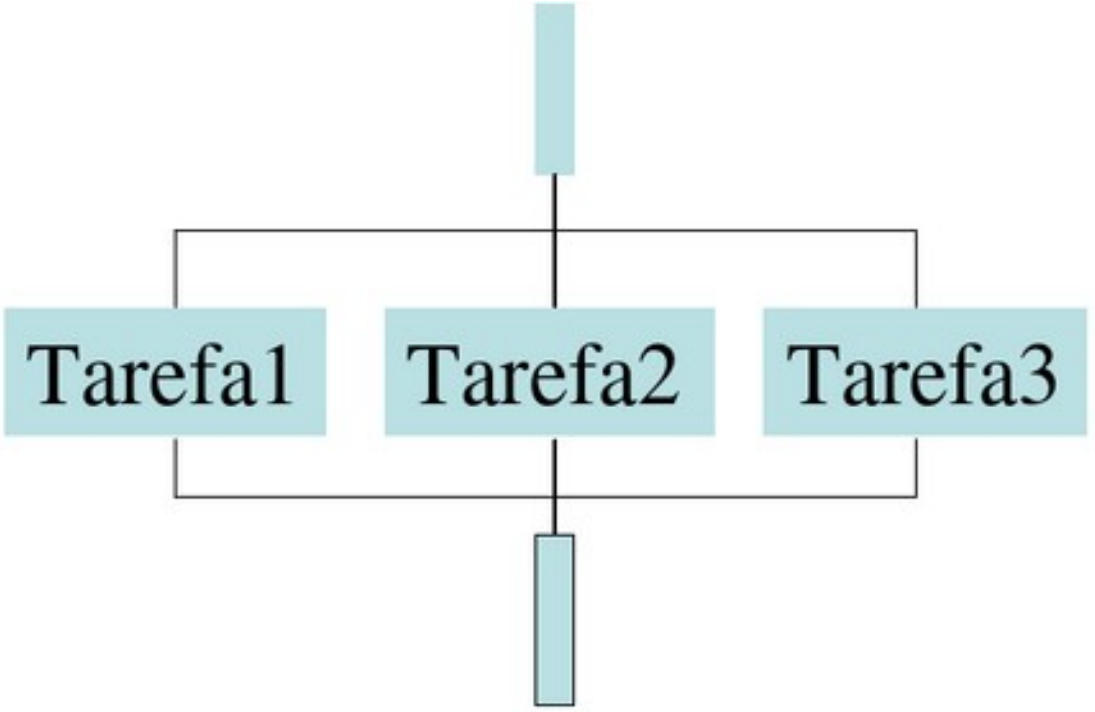
- Paradigma de programação que se concentra na execução (aparente) simultânea de múltiplas tarefas
- Conjunto de programas sequenciais ordinários que são executados em uma abstração de paralelismo
- Pode não ser executado simultaneamente
  - Ex.: SO alternando rapidamente (timeslicing)
- Melhora desempenho, responsividade, e aproveitamento de recursos

Como obter alto desempenho

- Aumento do desempenho do núcleo do processador
  - Aumento do clock ⇒ aumento de temperatura
  - Melhorias na arquitetura
- Processamento Paralelo
  - Múltiplas unidades de processamento
  - Paralelização da aplicação

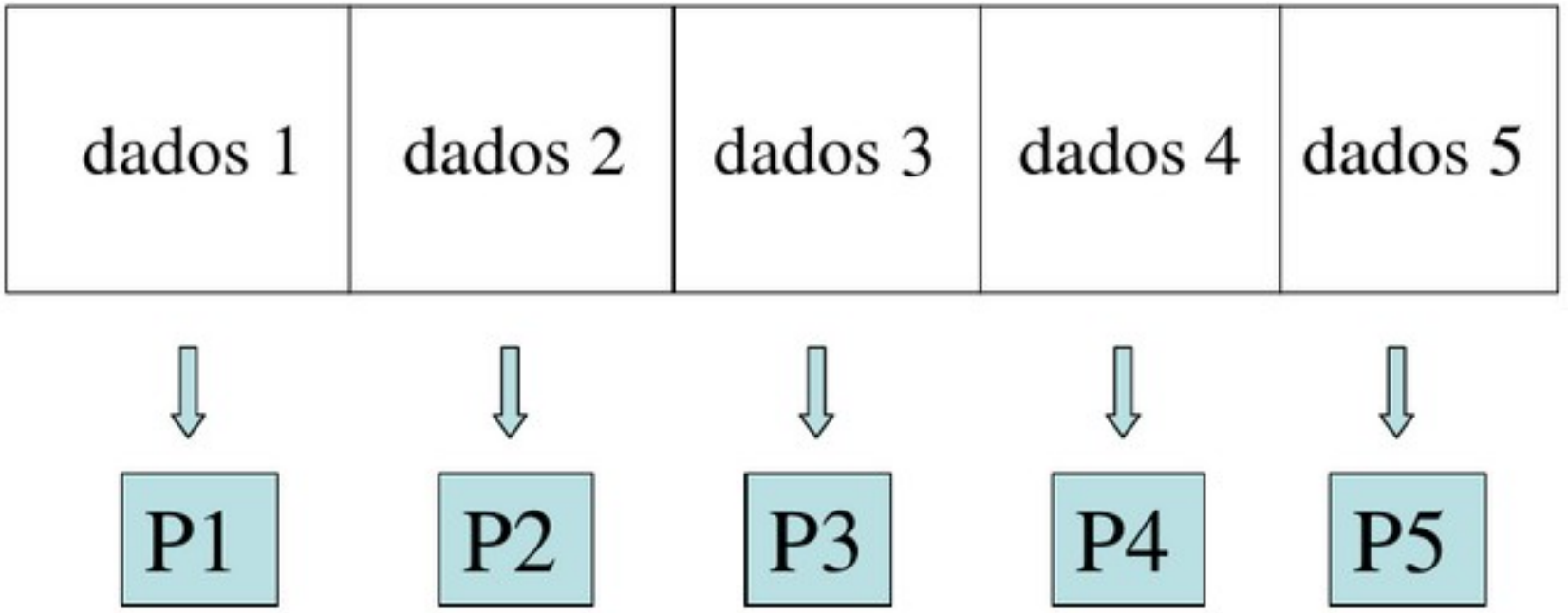
Processamento Paralelo

- Utilização de múltiplos processadores



Processamento Paralelo

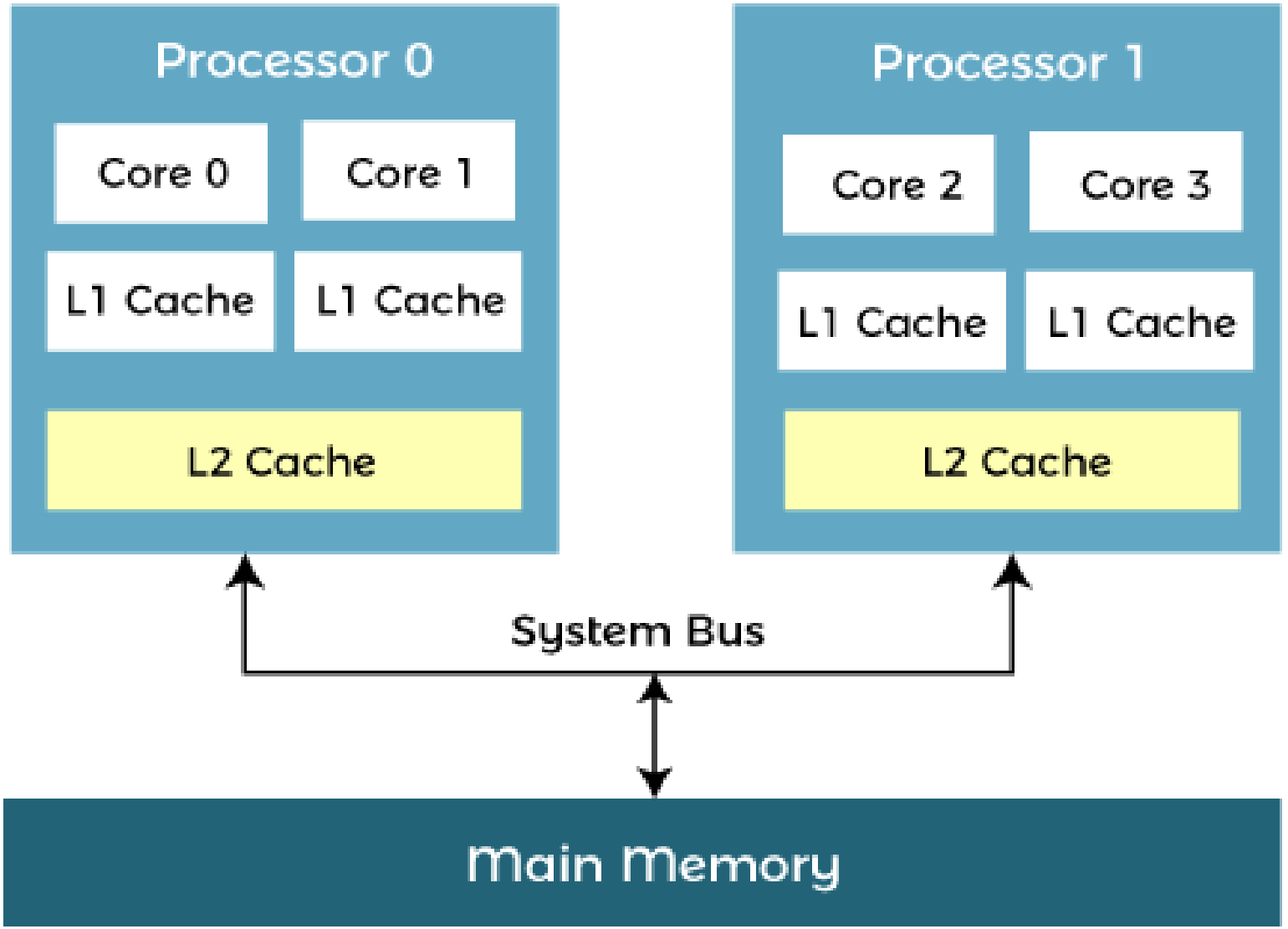
- Paralelismo de dados



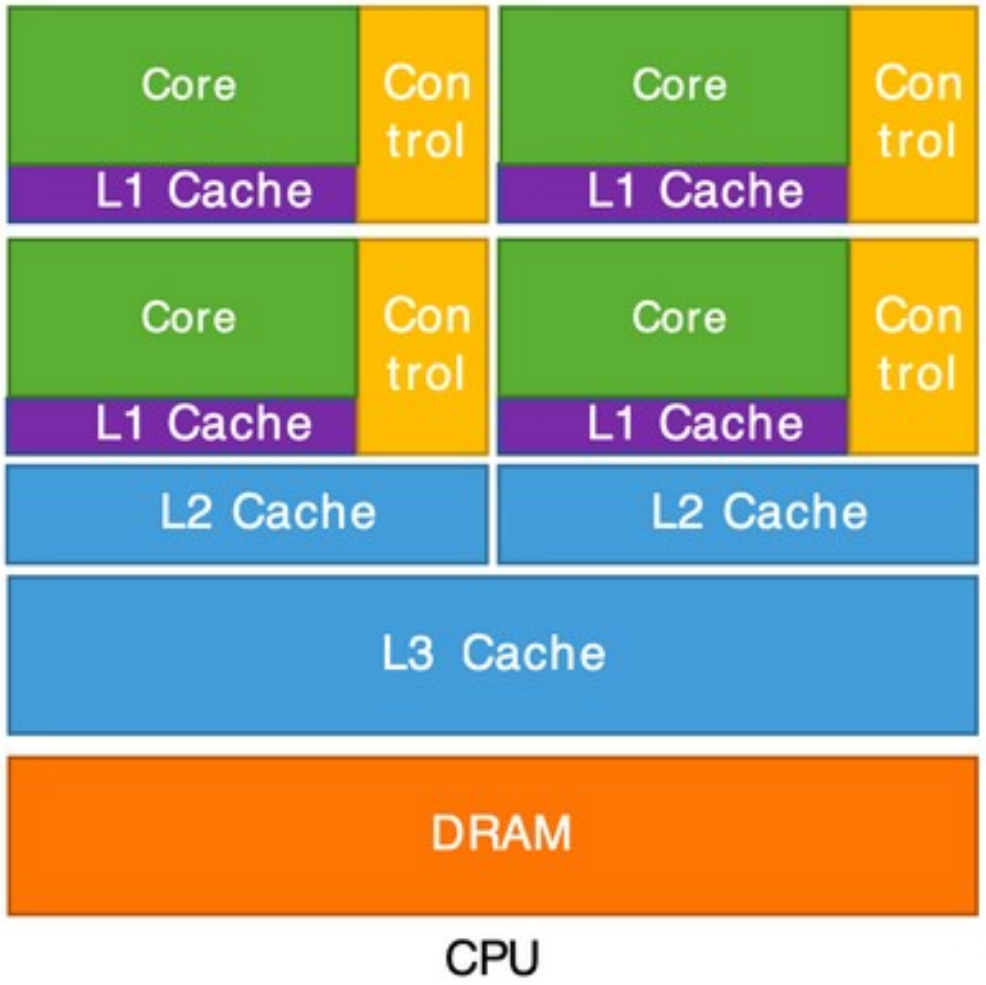
Hardware para Computação Paralela

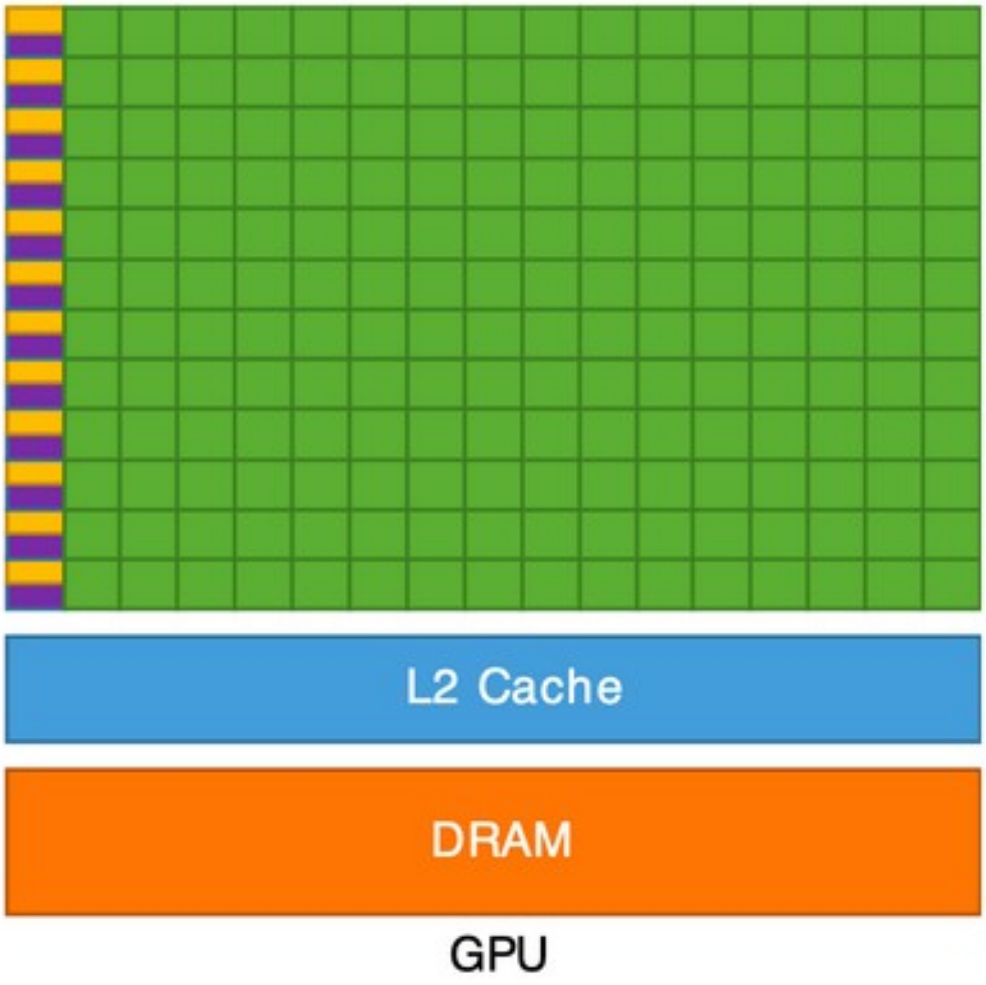
- CPUs Multi-core: Processadores com múltiplos núcleos para executar tarefas simultaneamente
  - Ex.: Intel Core i9, AMD Ryzen 9
- GPUs: Processadores com milhares de núcleos para processamento paralelo massivo
  - Ex.: NVIDIA GeForce RTX 4090, AMD Radeon RX 7900 XTX.

Multi-core


Processor 0
Core 0
Core 1
L1 Cache
L1 Cache
L2 Cache
Processor 1
Core 2
Core 3
L1 Cache
L1 Cache
L2 Cache
System Bus
Main Memory

CPU x GPU


Core
Con trol
L1 Cache
Core
Con trol
L1 Cache
Core
Con trol
L1 Cache
Core
Con trol
L1 Cache
L2 Cache
L2 Cache
L3 Cache
DRAM
CPU


L2 Cache
DRAM
GPU

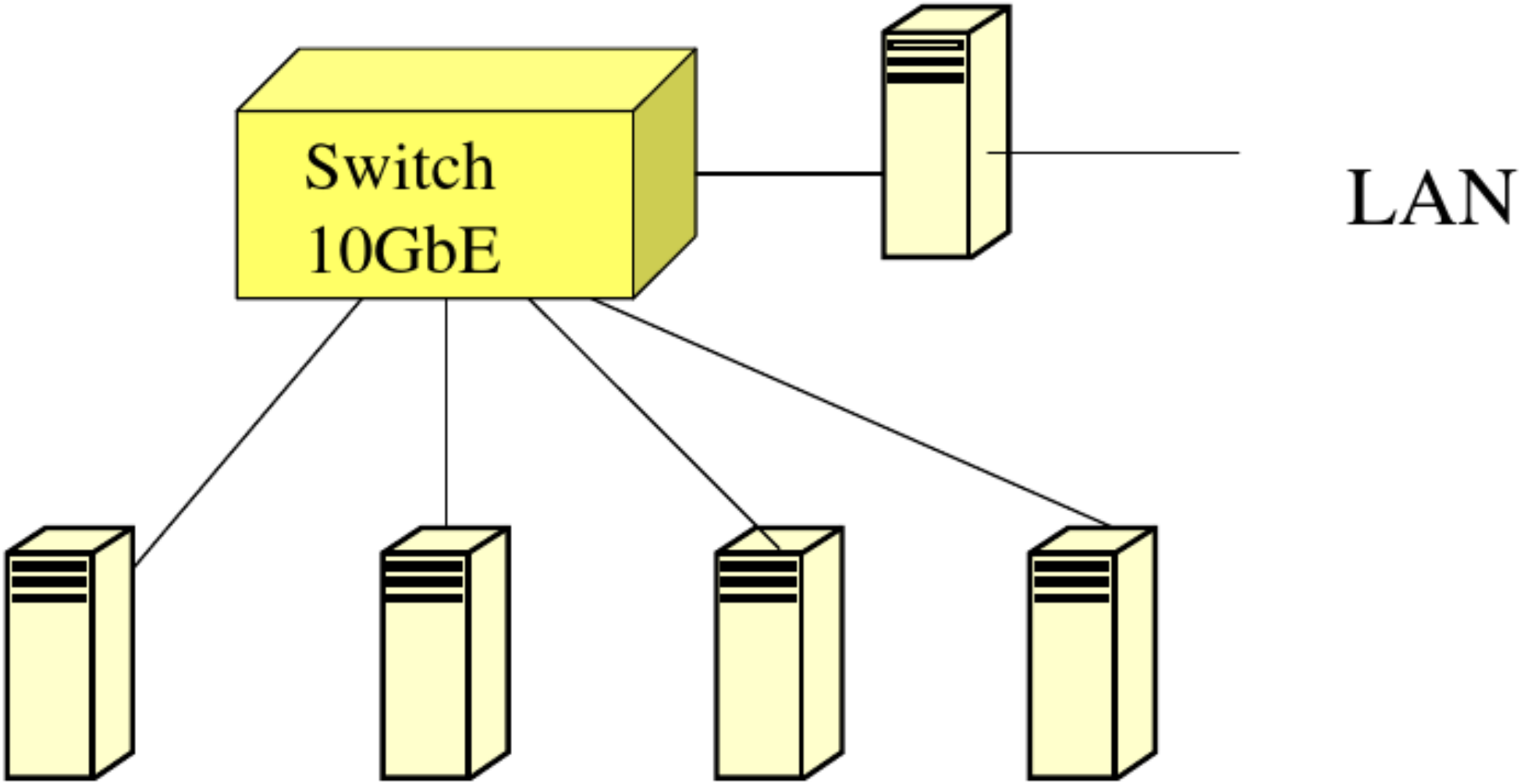
CPU x GPU


GIGABYTE
GEFORCE RTX 3050
GIGABYTE
GEFORCE
RTX
3050
GAMING OC 8G
OC

Arquiteturas Paralelas: Classificação

- Classificação de Flynn (SIMD/MIMD)
  - SIMD (Single Instruction, Multiple Data)
    - Múltiplos processadores executam a mesma instrução simultaneamente sobre diferentes conjuntos de dados
      - Processadores vetoriais, GPUs (para certas operações)
  - MIMD (Multiple Instruction, Multiple Data)
    - Múltiplos processadores executam instruções diferentes sobre diferentes conjuntos de dados
      - Multiprocessadores (Memória Compartilhada)
      - Multicomputadores (Memória Distribuída)

Cluster



Programação para Memória Compartilhada (Multicore/Multiprocessadores)

- Modelo
  - Threads (linhas de execução dentro de um processo)

- APIs e Bibliotecas
  - OpenMP: Diretivas e funções para paralelismo em C/C++ e Fortran. Simplifica a programação paralela em loops e outras regiões de código
  - Pthreads (POSIX Threads): API padrão POSIX para programação com threads em C/C++. Oferece maior controle, mas exige mais código manual

Programação para Memória Distribuída (Clusters/Multicomputadores)

- Modelo
  - Troca de mensagens entre processos independentes
- API e Padrão
  - MPI (Message Passing Interface): Padrão para comunicação entre processos em sistemas distribuídos
- Implementações de MPI
  - MPICH: Implementação popular e portável
  - Open MPI: Outra implementação amplamente utilizada, com bom desempenho e recursos avançados