

Programação Paralela

Prof. Rodrigo Martins

rodrigo.martins@francomontoro.com.br



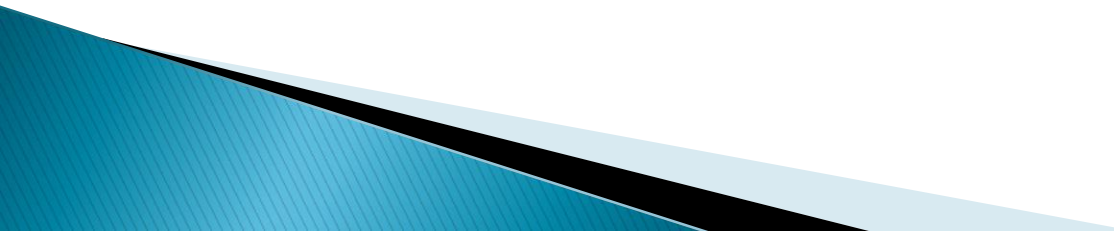
Cronograma da Aula

- ▶ Modelos de computação paralela e suas arquiteturas
- ▶ Exercícios

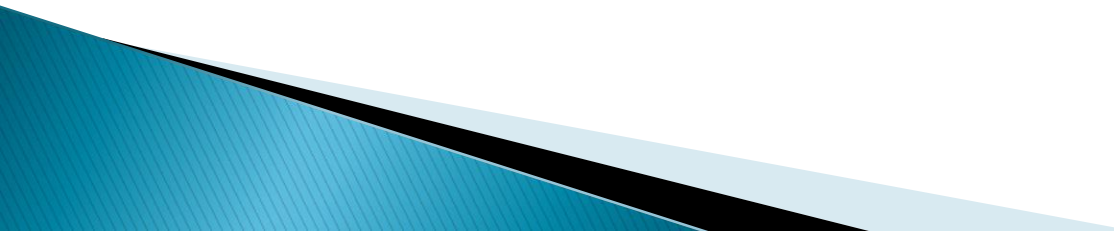
Modelos de computação paralela

- Modelos de computação paralela amplamente utilizados:
 - Modelo de memória compartilhada;
 - Modelo de memória distribuída;
 - Modelo híbrido;

Modelo de memória compartilhada

- No modelo de memória compartilhada, vários processadores ou núcleos compartilham uma única área de memória principal.
 - Isso permite que os processadores acessem e modifiquem a mesma área de memória, permitindo que eles cooperem e coordenem seus esforços.
- 

Modelo de memória compartilhada

- Em um sistema de memória compartilhada, as operações de leitura e escrita na memória compartilhada são feitas com mais eficiência, pois os processadores podem acessá-la diretamente sem a necessidade de comunicação explícita.
 - Isso pode melhorar significativamente o desempenho em certos tipos de aplicativos, como processamento de imagens, edição de vídeo, simulações científicas e jogos.
- 

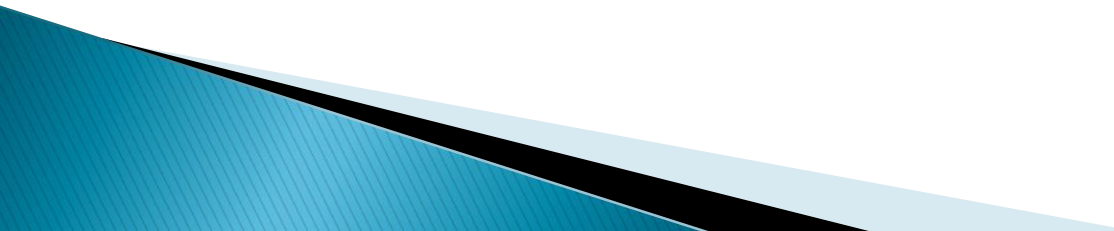
Modelo de memória compartilhada

- **Desafios:**

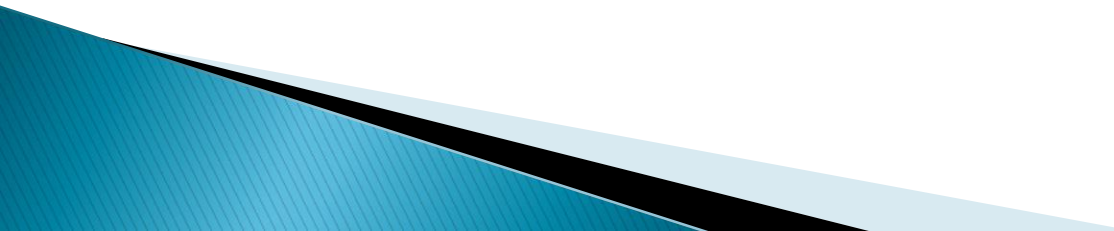
- Se vários processadores tentarem acessar e modificar a mesma área de memória ao mesmo tempo, pode ocorrer um problema conhecido como "conflito de memória".
- Isso pode resultar em um estado inconsistente e produzir resultados incorretos. Para evitar esses problemas, os sistemas de memória compartilhada usam vários mecanismos, como bloqueios e semáforos, para sincronizar o acesso à memória compartilhada.

Modelo de memória compartilhada

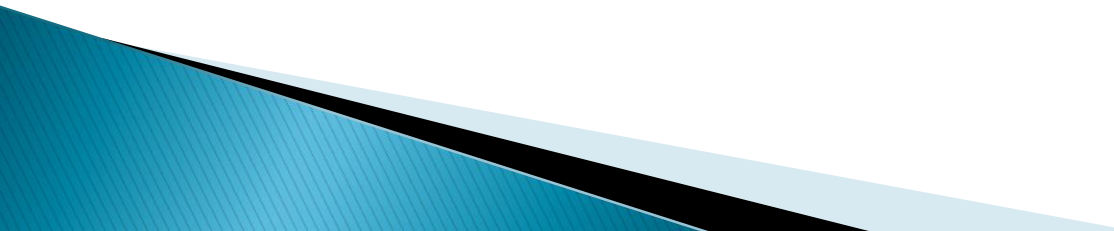
- Exemplos:

- Multiprocessadores e multicores, onde vários processadores ou núcleos são colocados no mesmo chip.
 - As linguagens de programação com suporte para memória compartilhada incluem C/C++, Java e Python, que incluem bibliotecas específicas, como OpenMP e PThreads, para programação paralela usando memória compartilhada.
- 

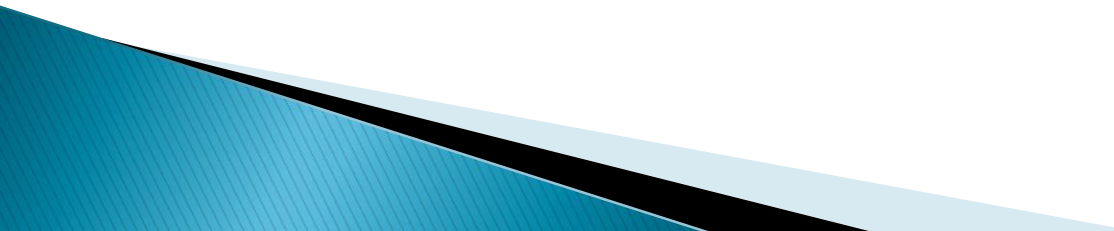
Modelo de memória distribuída

- No modelo de memória distribuída, cada processador ou nó de processamento possui sua própria memória local e os nós se comunicam através de trocas explícitas de mensagens.
 - Neste modelo, a comunicação entre os nós é feita através da troca de mensagens explícitas, que são enviadas de um nó para outro pela rede.
- 

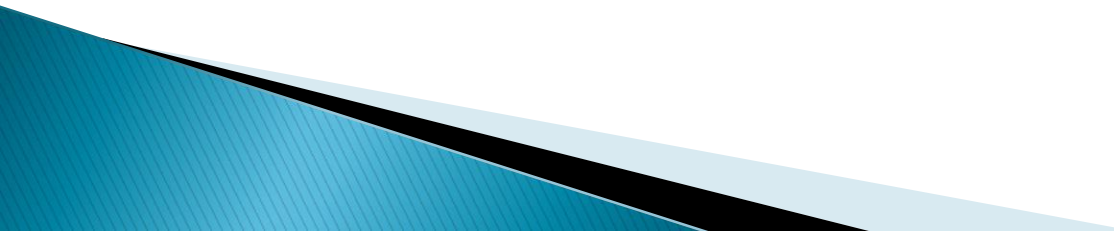
Modelo de memória distribuída

- Cada nó tem acesso apenas à sua própria memória local, o que significa que os processadores precisam explicitamente enviar dados uns aos outros para compartilhar informações.
 - Esse modelo é adequado para sistemas que precisam de muita escalabilidade, com centenas ou milhares de processadores trabalhando juntos.
- 

Modelo de memória distribuída

- Uma representação visual do modelo de memória distribuída seria uma imagem de uma rede de computadores conectados por cabos, onde cada computador funciona como um nó de processamento com sua própria memória local.
 - Cada nó pode executar tarefas independentes ou colaborar com outros nós para processar grandes conjuntos de dados ou executar simulações científicas em paralelo.
 - A comunicação entre os nós ocorre por meio de trocas explícitas de mensagens, que são transmitidas pela rede.
- 

Modelo de memória distribuída

- Em sistemas de memória distribuída, a comunicação entre os nós pode ter um gargalo de desempenho, especialmente em redes de computadores lentas.
 - Para melhorar o desempenho, muitas linguagens de programação de memória distribuída incluem bibliotecas para gerenciamento de comunicação, como MPI (Message Passing Interface) e PVM (Parallel Virtual Machine), que facilitam a troca de mensagens entre os nós.
- 

Modelo de memória distribuída

- Exemplos

- Grandes simulações científicas, processamento de dados em larga escala e sistemas de processamento de dados distribuídos, como o Apache Hadoop e o Spark.

Modelo de memória distribuída

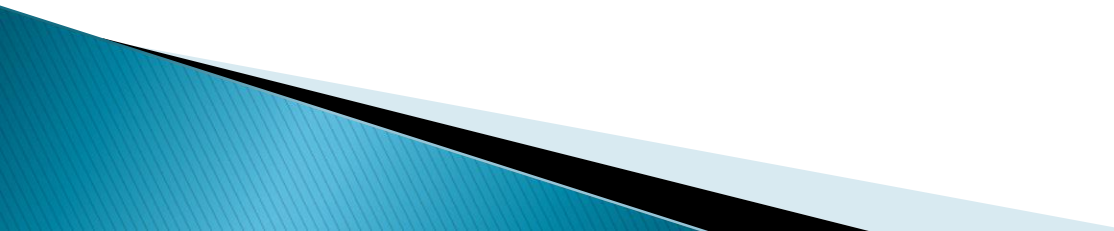
- Uma das principais **vantagens** do modelo de memória distribuída é sua **escalabilidade**.
- À medida que mais nós são adicionados ao sistema, a capacidade de processamento pode ser aumentada quase linearmente.

Modelo de memória distribuída

- **Desafios**

- O uso de múltiplos nós pode apresentar desafios adicionais de sincronização e gerenciamento de recursos, o que pode tornar a programação mais complexa do que em sistemas de memória compartilhada.

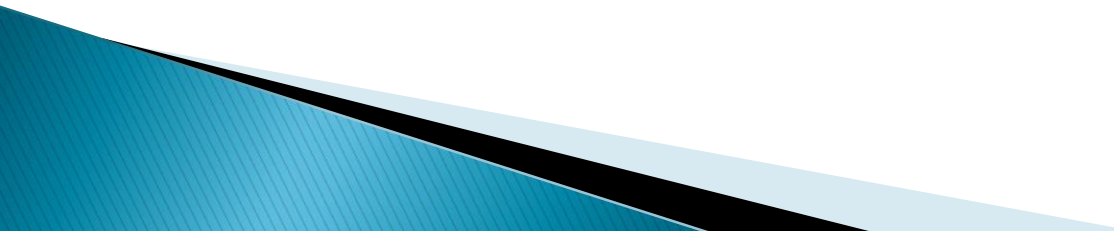
Modelo híbrido

- É uma abordagem que combina elementos do modelo de memória compartilhada e do modelo de memória distribuída.
 - Nesse modelo, vários nós de processamento estão interconectados em uma rede, com cada nó tendo um ou mais processadores e uma memória local.
- 

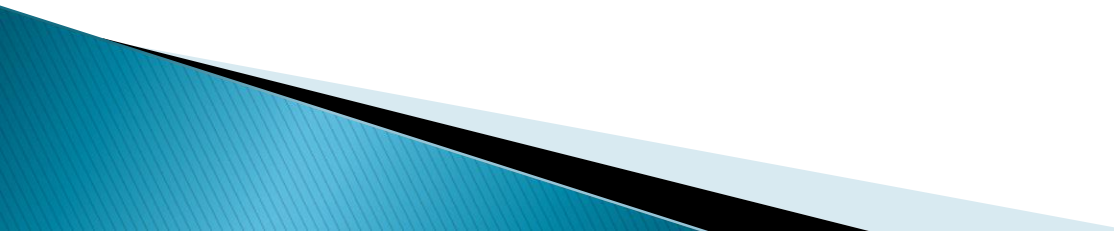
Modelo híbrido

- Os nós de processamento são agrupados em clusters e cada cluster tem uma memória compartilhada.
- Os clusters são então interconectados em uma rede de comunicação de alta velocidade, permitindo que os processos sejam distribuídos entre os nós de processamento.

Modelo híbrido

- Essa abordagem oferece a flexibilidade e a escalabilidade do modelo de memória distribuída, juntamente com a eficiência do modelo de memória compartilhada, já que os processos dentro de um cluster podem se comunicar e compartilhar memória de forma mais rápida e eficiente do que entre clusters.
 - O modelo híbrido também é útil em situações em que a aplicação pode ser dividida em partes que se beneficiam do uso de um ou outro modelo.
- 

Modelo híbrido

- **Exemplo:**
 - O modelo MPI+OpenMP, combina o uso de OpenMP, uma API para programação de memória compartilhada, com MPI, um protocolo de comunicação de memória distribuída.
 - O MPI é usado para comunicar entre os nós do cluster, enquanto o OpenMP é usado para paralelizar os processos dentro de cada nó de processamento.
 - Isso permite uma abordagem escalonável e eficiente para a programação paralela em ambientes de cluster.
- 

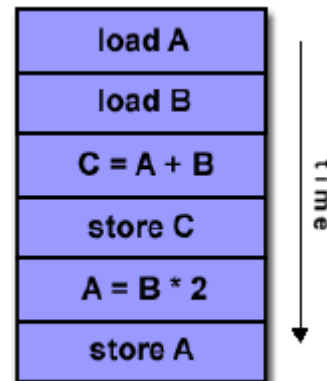
Taxonomia de Flynn

- Uma das metodologias mais conhecidas e utilizadas para classificar a arquitetura de um computador ou conjunto de computadores é a taxonomia de Flynn (1966).
- Esta metodologia classifica a arquitetura dos computadores segundo duas dimensões independentes: **instruções** e **dados**, em que cada dimensão pode tomar apenas um de dois valores distintos: **single** ou **multiple**.

	<i>Single Data</i>	<i>Multiple Data</i>
<i>Single Instruction</i>	SISD <i>Single Instruction Single Data</i>	SIMD <i>Single Instruction Multiple Data</i>
<i>Multiple Instruction</i>	MISD <i>Multiple Instruction Single Data</i>	MIMD <i>Multiple Instruction Multiple Data</i>

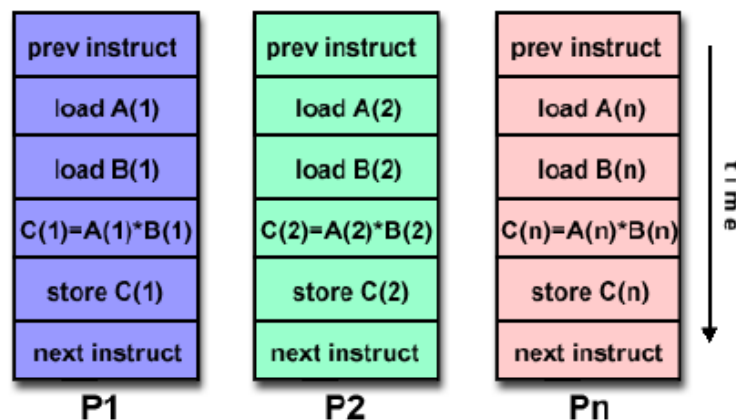
SISD – Single Instruction Single Data

- Corresponde à arquitetura dos computadores com um único processador.
 - Apenas uma instrução é processada a cada momento.
 - Apenas um fluxo de dados é processado a cada momento.
- Exemplos: PCs, workstations e servidores com um único processador.



SIMD – Single Instruction Multiple Data

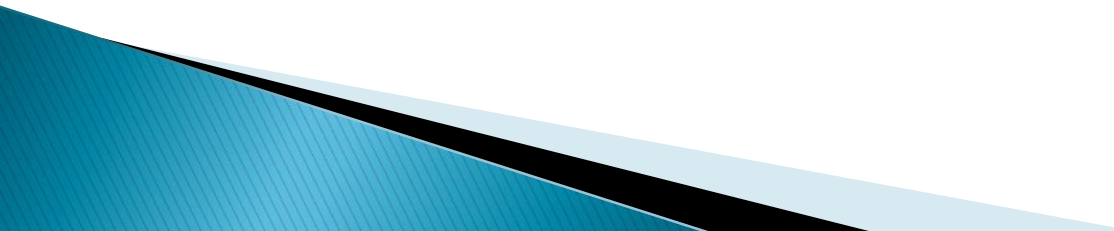
- SIMD (Single Instruction, Multiple Data) é uma abordagem de computação paralela que permite a execução simultânea de uma única instrução em diferentes conjuntos de dados.
- Um único processador executa a mesma operação em diferentes dados simultaneamente, o que é útil para tarefas que envolvem processamento de grandes quantidades de dados em paralelo, como processamento de imagens e áudio.



Arquitetura – SIMD

- É especialmente útil em tarefas que exigem o mesmo tipo de processamento em muitos dados, como adicionar ou multiplicar constantes em matrizes grandes ou aplicar um filtro em uma imagem.
- Nesse caso, um único processador pode executar a mesma operação em todos os dados ao mesmo tempo, o que reduz o tempo total de processamento.

Arquitetura – SIMD

- É implementado em muitos processadores modernos, incluindo processadores gráficos (GPUs) e processadores de sinais digitais (DSPs).
 - Esses processadores possuem unidades de processamento vetorial que permitem a execução simultânea de várias operações em um conjunto de dados, resultando em uma melhoria significativa na velocidade de processamento em relação a um processador convencional.
- 

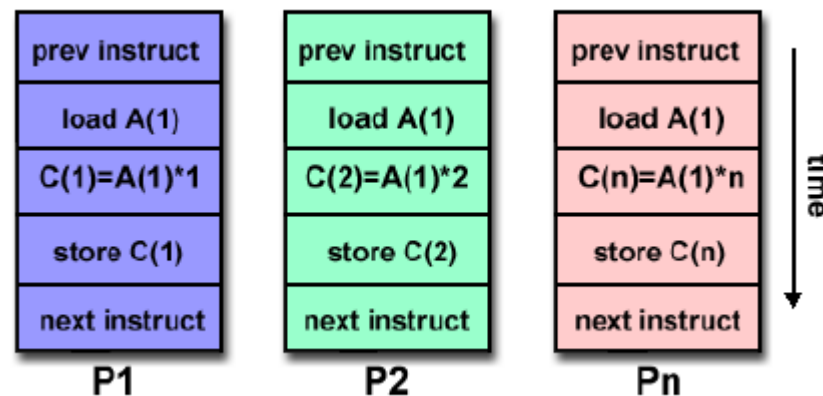
Arquitetura – SIMD

- Limitações

- Ele só pode ser usado para tarefas em que a mesma operação é executada em todos os dados, o que significa que não é adequado para tarefas mais complexas que requerem diferentes tipos de operações em diferentes conjuntos de dados. Além disso, a eficiência do modelo SIMD depende da estrutura dos dados e da capacidade do compilador de otimizar o código para aproveitar totalmente as capacidades do processador SIMD.

MISD – Multiple Instruction Single Data

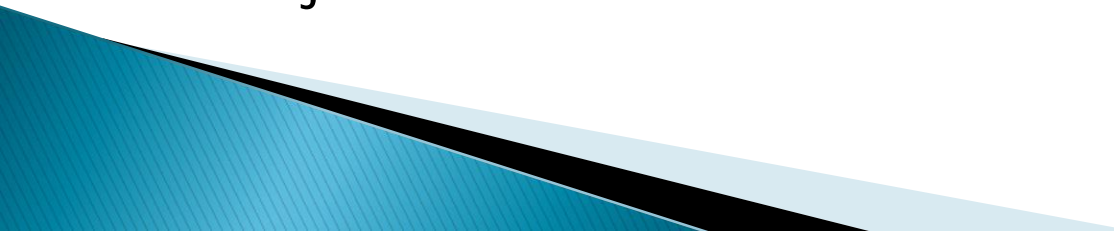
- Tipo de arquitetura paralela desenhada para problemas específicos caracterizados por um alto padrão de regularidade funcional (e.g. processamento de sinal).
- Constituída por uma pipeline de unidades de processamento independentes que operam sobre um mesmo fluxo de dados enviando os resultados em uma unidade para a próxima.
- Cada unidade de processamento executa instruções diferentes a cada momento.



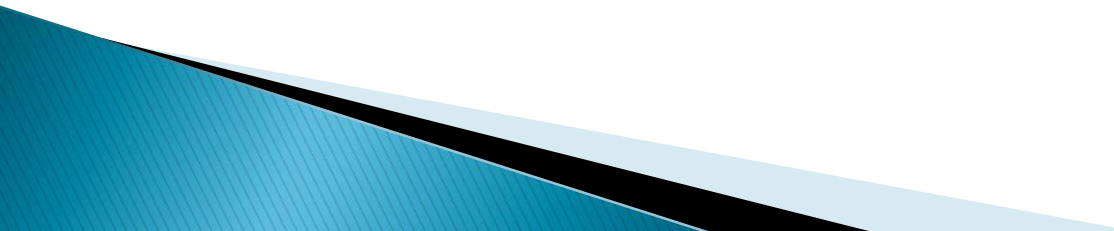
MISD – Multiple Instruction Single Data

- A arquitetura **MISD** é usada em sistemas que exigem alto nível de redundância, como sistemas críticos de segurança, onde vários processadores executam simultaneamente tarefas de monitoramento e controle em um único conjunto de dados.

MIMD – Multiple Instruction Multiple Data

- **MIMD (Multiple Instruction, Multiple Data)** é uma abordagem de computação paralela que permite que cada processador execute uma tarefa diferente em um conjunto de dados diferente.
 - Diferentemente do modelo SIMD, em que todos os processadores executam a mesma instrução em diferentes conjuntos de dados, o modelo MIMD permite que cada processador execute diferentes instruções em diferentes conjuntos de dados.
- 

Arquitetura – MIMD

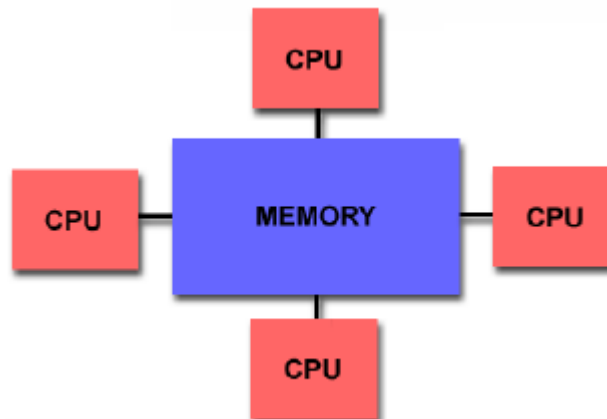
- É amplamente utilizada em sistemas de alto desempenho, como clusters de servidores, sistemas de banco de dados distribuídos e simulação de sistemas complexos.
 - Nessa arquitetura, cada processador possui sua própria memória local e pode executar um programa diferente, permitindo que várias tarefas sejam executadas simultaneamente em diferentes conjuntos de dados.
- 

Arquitetura – MIMD

- Tipo de arquitetura paralela predominante atualmente.
 - Cada unidade de processamento executa instruções diferentes a cada momento.
 - Cada unidade de processamento pode operar sobre um fluxo de dados diferente.
- Exemplos: **multiprocessadores e multicomputadores.**

Multiprocessadores

- Um multiprocessador é um computador em que todos os processadores partilham o acesso à memória física.
- Os processadores executam de forma independente mas o espaço de endereçamento global é partilhado.
- Qualquer alteração sobre uma posição de memória realizada por um determinado processador é igualmente visível por todos os restantes processadores.

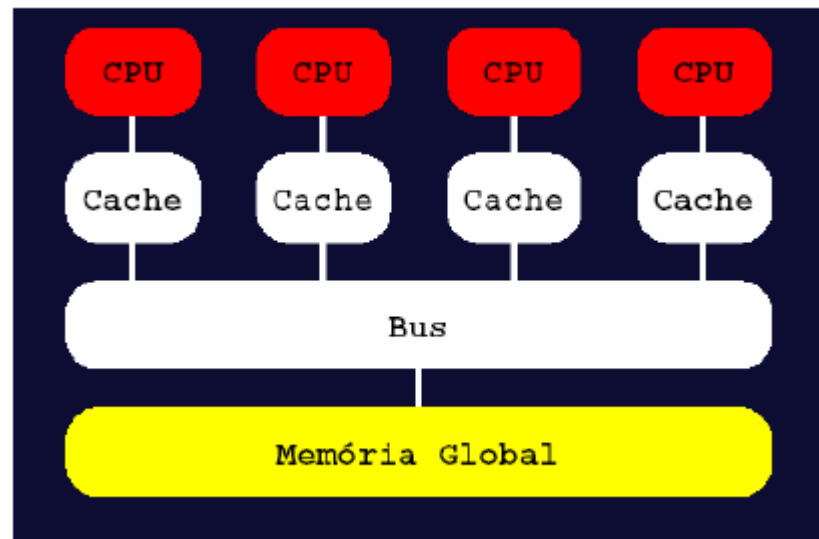


Multiprocessadores

- Existem duas grandes classes de multiprocessadores:
 - Uniform Memory Access Multiprocessor (UMA) ou Cache Coherent Uniform Memory Access Multiprocessor (CC-UMA) ou Symmetrical Multiprocessor (SMP) ou Centralized Multiprocessor
 - Non-Uniform Memory Access Multiprocessor (NUMA) ou Distributed Multiprocessor

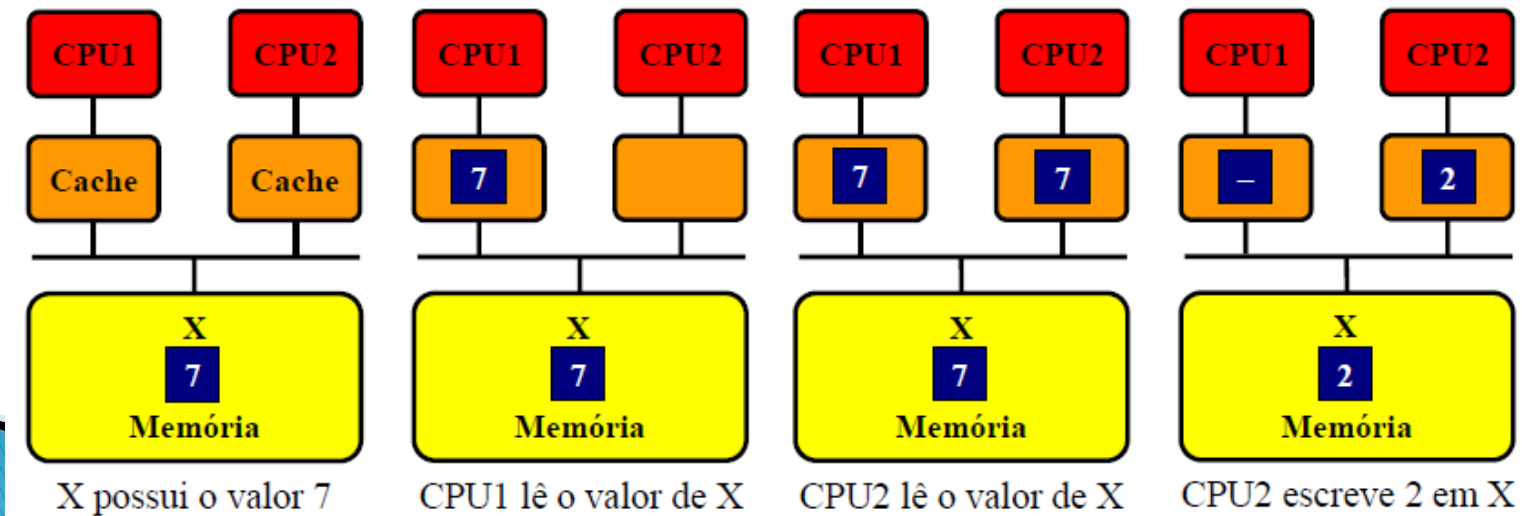
Multiprocessadores

- Uniform Memory Access Multiprocessor (UMA)
 - Todos os processadores têm tempos de acesso idênticos a toda a memória.
 - A coerência das caches é implementada pelo hardware (**write invalidate protocol**).



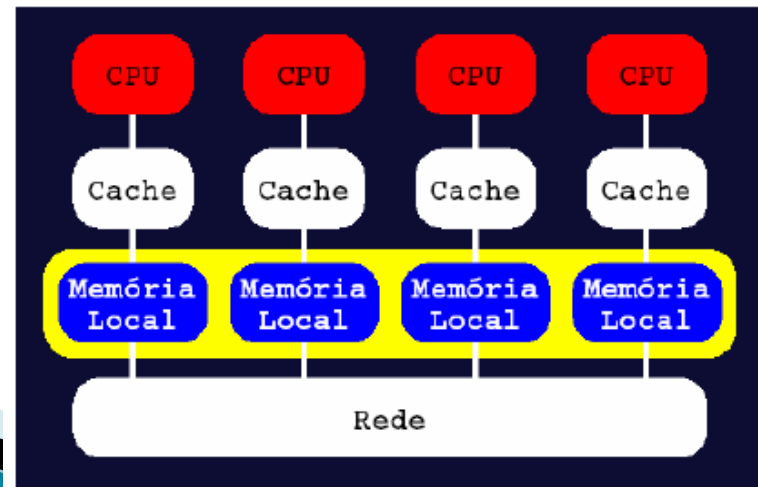
Write Invalidate Protocol

- Antes de se escrever um valor em memória, todas as cópias existentes nas caches dos outros processadores são invalidadas. Quando mais tarde, esses outros processadores tentam ler o valor invalidado, acontece um cache miss o que os obriga a atualizar o valor a partir da memória.



Multiprocessadores


- Non-Uniform Memory Access Multiprocessor (NUMA)
 - Os processadores têm tempos de acesso diferentes a diferentes áreas da memória.
 - Se a coerência das caches for implementada pelo hardware (**directory-based protocol**) são também designados por Cache Coherent NUMA (CC-NUMA).



Directory-Based Protocol

- Associado a cada processador existe um diretório com informação sobre o estado dos seus blocos de memória. Cada bloco pode estar num dos seguintes estados:
 - **Uncached:** não está na cache de nenhum processador.
 - **Shared:** encontra-se na cache de um ou mais processadores e a cópia em memória está correta.
 - **Exclusive:** encontra-se apenas na cache de um processador e a cópia em memória está obsoleta.

Multiprocessadores

- Vantagens e inconvenientes:
 - (+) Partilha de dados entre tarefas é conseguida de forma simples, uniforme e rápida.
 - (-) Necessita de mecanismos de sincronização para obter um correto manuseamento dos dados.
 - (-) Pouco escalável. O aumento do número de processadores aumenta a contenção no acesso à memória e torna inviável qualquer mecanismo de coerência das caches.
 - (-) Custo elevado. É difícil e bastante caro desenhar e produzir computadores cada vez com um maior número de processadores.
- 

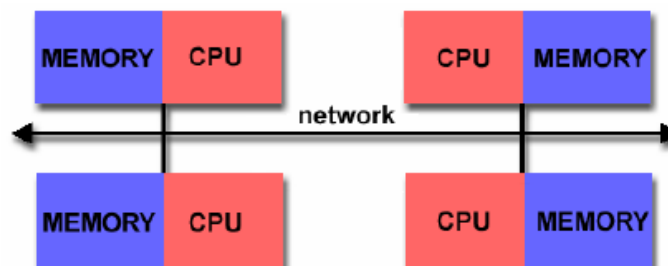
Multicomputadores

- Um multicomputador é um conjunto de computadores ligados por rede em que cada computador tem acesso exclusivo à sua memória física.
- O espaço de endereçamento de cada computador não é partilhado pelos restantes computadores, ou seja, não existe o conceito de espaço de endereçamento global.
- As alterações sobre uma posição de memória realizada por um determinado processador não são visíveis pelos processadores dos restantes computadores, ou seja, não existe o conceito de coerência das caches.

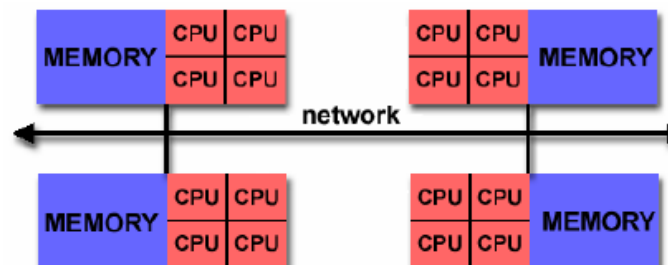
Multicomputadores

- Existem duas grandes classes de multicomputers:
 - Distributed Multicomputer
 - Distributed-Shared Multicomputer.

■ *Distributed Multicomputer*



■ *Distributed-Shared Multicomputer*

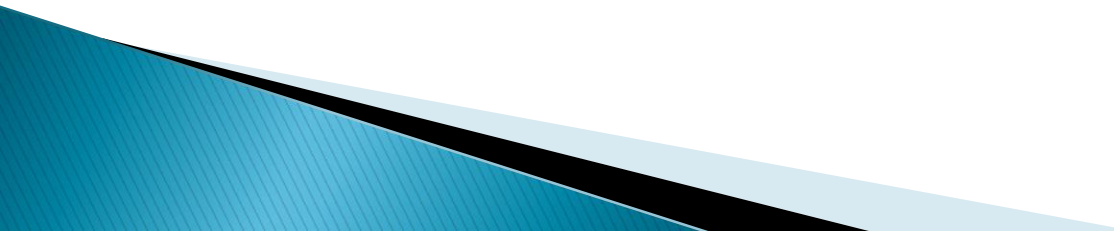


Multicomputadores

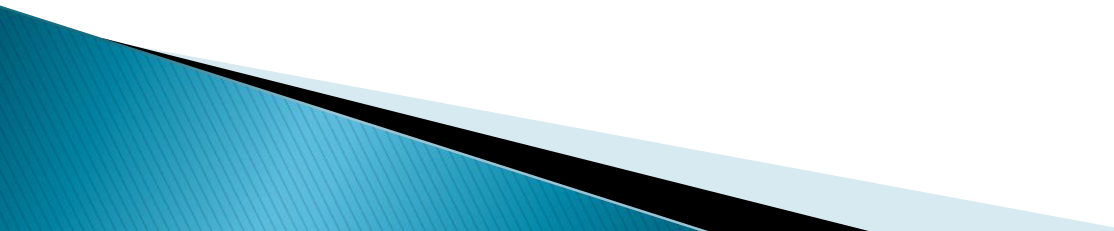
- **Vantagens e inconvenientes:**

- (+) O aumento do número de computadores aumenta proporcionalmente a memória disponível sem necessitar de mecanismos de coerência das caches.
- (+) Fácil escalabilidade a baixo custo. O aumento do poder de computação pode ser conseguido à custa de computadores de uso doméstico.
- (-) Necessita de mecanismos de comunicação para partilha de dados entre tarefas de diferentes computadores.
- (-) O tempo de acesso aos dados entre diferentes computadores não é uniforme e é por natureza mais lento.
- (-) Pode ser difícil converter estruturas de dados previamente existentes para memória partilhada em estruturas de dados para memória distribuída.

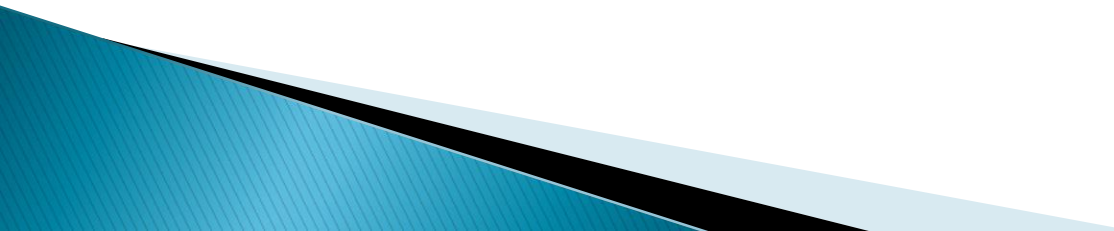
Exercícios

1. Explique o que é o modelo de memória compartilhada. Quais são as principais características desse modelo e em que tipo de arquitetura de computador ele é mais utilizado?
 2. Descreva o modelo de memória distribuída. Quais são as principais características desse modelo e em que tipo de arquitetura de computador ele é mais utilizado?
- 

Exercícios

3. Escreva as diferenças entre os modelos de memória compartilhada e memória distribuída. Quais são as vantagens e desvantagens de cada modelo e em que situações cada um é mais adequado?
 4. Explique o que é o modelo híbrido de programação paralela. Como esse modelo combina elementos dos modelos de memória compartilhada e memória distribuída? Quais são as principais vantagens desse modelo e em que situações ele é mais adequado?
- 

Exercícios

5. Compare o modelo SIMD e o modelo MIMD em termos de suas abordagens de computação paralela. Discuta suas semelhanças, diferenças e situações em que um modelo é mais adequado do que o outro.
 6. Considere uma tarefa de processamento de imagem que envolve a aplicação de um filtro em uma imagem grande. Discuta como o modelo SIMD e o modelo MIMD podem ser usados para executar essa tarefa em paralelo e quais são as vantagens e desvantagens de cada modelo.
- 

Exercícios

7. Descreva a arquitetura MIMD, suas aplicações e desafios.
8. Analise a arquitetura MISD e explique seu funcionamento.

Referências desta Aula

- DE ROSE, César A. F.; NAVAUX, Philippe O. A. Arquiteturas Paralelas. Porto Alegre: Sagra-Luzzato, 2008.
- Aula montada com base no material do professor Ricardo Rocha – DCC-FCUP. Disponível em:
<https://www.dcc.fc.up.pt/~ricroc/aulas/0708/ppd/apontamentos/fundamentos.pdf>

FIM
Obrigado!

Rodrigo