

Tópicos em Computação de Alto Desempenho

Aula 01: Arquitetura de Sistemas Paralelos e Distribuídos

Prof. Me. Edjalma Queiroz da Silva

Por que Computação Paralela e Distribuída?

- ▶ Sistemas de computadores seqüenciais cada vez mais velozes
 - ▶ velocidade de processador
 - ▶ memória
 - ▶ comunicação com o mundo externo
- ▶ Quanto mais se tem, mais se quer.....
- ▶ Demanda computacional está aumentando cada vez mais: visualização, base de dados distribuída, simulações, etc.
- ▶ limites em processamento seqüencial
- ▶ Restritas a um único processamento de dados.

Por que Computação Paralela e Distribuída?

- ▶ que tal utilizar vários processadores?
- ▶ dificuldades encontradas
- ▶ mas como?
- ▶ paralelizar uma solução?
- ▶ O que pode ser paralelizado em um programa?

Existem vários desafios em Computação Paralela e Distribuída

Definindo melhor alguns conceitos...

•Concorrência

- termo mais geral, um programa pode ser constituído por mais de um thread/processo concorrendo por recursos

•Paralelismo

- uma aplicação é executada por um conjunto de processadores em um ambiente único (dedicados)

•Computação distribuída

- aplicações sendo executadas em plataformas distribuídas

Definindo melhor alguns conceitos...

•O que é um Programação Concorrente?

•Um programa concorrente é um **conjunto de programas sequenciais ordinários** os quais são executados em uma abstração de **paralelismo**.

•**Paralelismo** é o uso simultâneo de múltiplos recursos computacionais para resolver um **problema**.

•Para tal, o **problema deve ser capaz de:**

- Ser quebrado em diversas partes para ser resolvido simultaneamente.
- Executar múltiplas instruções computacionais a qualquer momento.
- Ser resolvido em menos tempo com múltiplos recursos computacionais do que com um único recurso.

Definindo melhor alguns conceitos...

Qualquer que seja o conceito, o que queremos?

- estabelecer a solução do problema
- lidar com recursos independentes
- aumentar desempenho e capacidade de memória
- fazer com que usuários e computadores trabalhem em espírito de colaboração

O que paralelizar?

- ▶ Concorrência pode estar em diferentes níveis de sistemas computacionais atuais
 - ▶ hardware
 - ▶ Sistema Operacional
 - ▶ Aplicação
- ▶ As principais questões que são focadas são:
 - ▶ Desempenho
 - ▶ Corretude
 - ▶ possibilidade de explorar o paralelismo

Por que paralelizar?

Aplicação Paralela

- ▶ várias tarefas
- ▶ vários processadores
- ▶ redução no tempo total de execução
- ▶ Metas
- ▶ Aumento no desempenho
- ▶ Maior eficiência

Por que paralelizar?

Aplicação Paralela

- ▶ várias tarefas
- ▶ vários processadores
- ▶ redução no tempo total de execução
- ▶ Metas
- ▶ Aumento no desempenho
- ▶ Maior eficiência

Introdução a Classificação de Máquinas

•Por que estudar classificações ?

-Para identificar critérios da classificação

•Por que os critérios são importantes e quais são as suas implicações

-Para analisar todas possibilidades e efeitos destas possibilidades nas arquiteturas

•Mesmo para classes que não foram implementadas ou implementações que não deram certo

-Para entender como se deu a evolução da área

-Para planejar a evolução da área

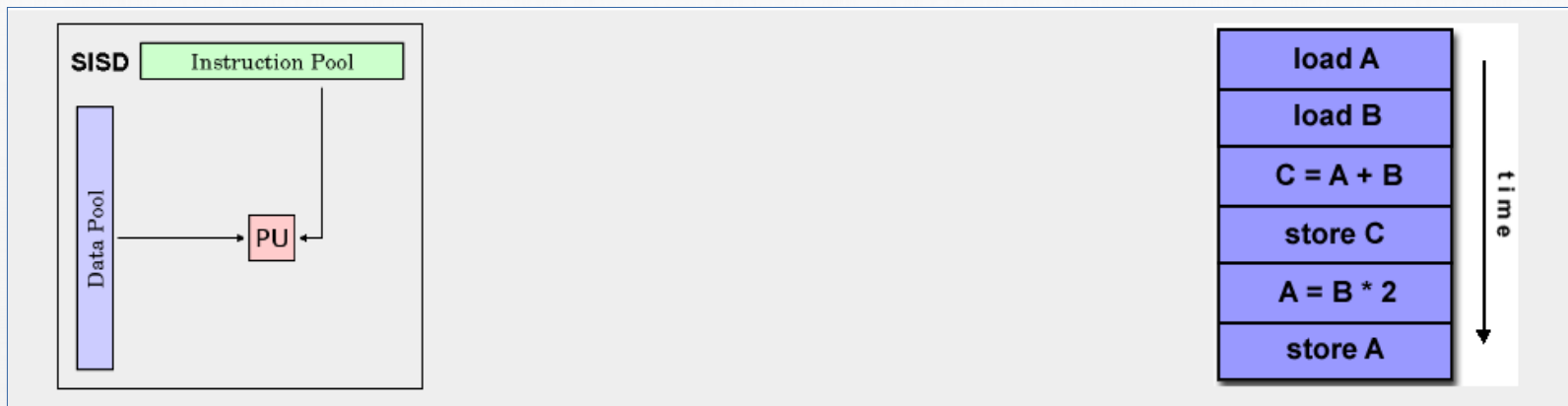
Classificação de Flynn

- Origem em meados dos anos 70
- Máquinas classificadas por **fluxo de dados** e **fluxo de instruções**

	SD (<i>Single Data</i>)	MD (<i>Multiple Data</i>)
SI (<i>Single Instruction</i>)	SISD Máquinas von Neumann convencionais	SIMD Máquinas <i>Array</i> (CM-2, MasPar)
MI (<i>Multiple Instruction</i>)	MISD Sem representante (até agora)	MIMD Multiprocessadores e multicomputadores (nCUBE, Intel Paragon, Cray T3D)

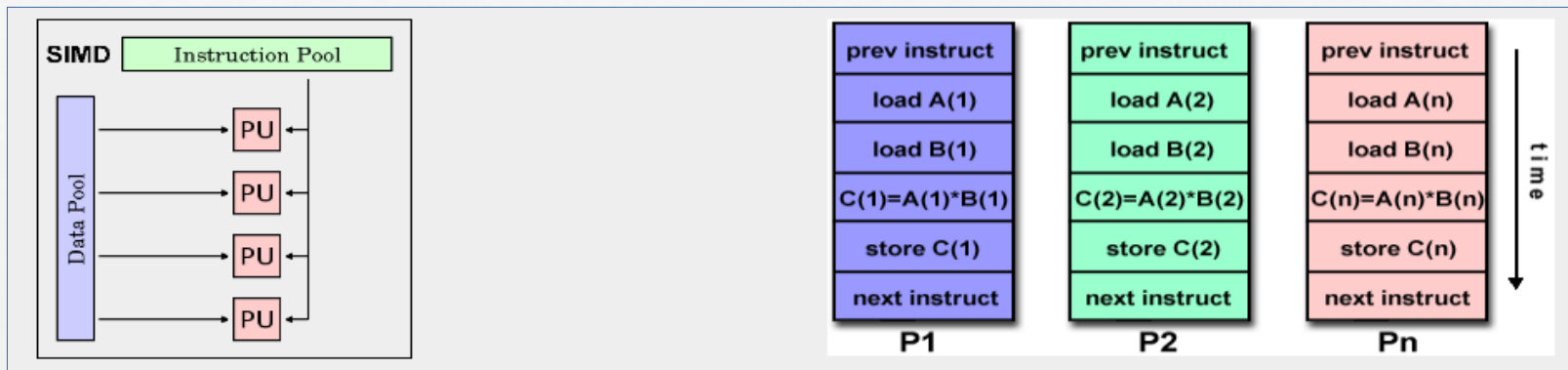
SISD (Single Instruction Single Data)

- Um único fluxo de instruções atua sobre um único fluxo de dados
- Nessa classe, são enquadradas:
 - Máquinas Von Neumann tradicionais com apenas um processador



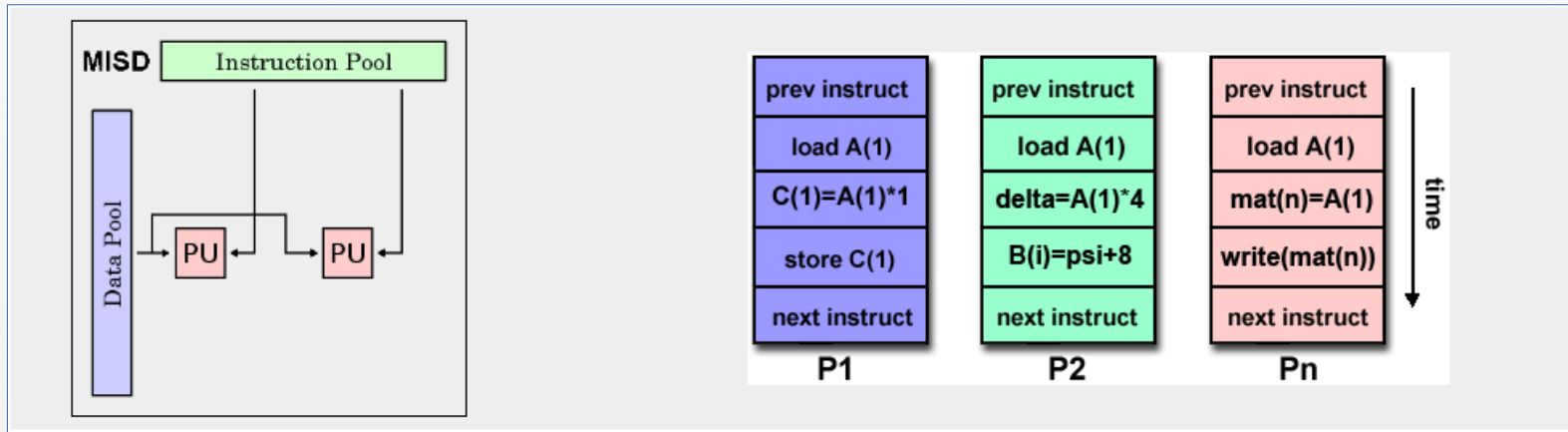
SIMD (Single Instruction Multiple Data)

- Processamento controlado por 1 unidade de controle, alimentada por 1 fluxo de instruções
- Mesma instrução é enviada para diversos processadores
- Todos os processadores executam instruções em paralelo, de forma síncrona, sobre diferentes fluxos de dados
- Único programa executado sobre diferentes dados
- Assemelhe-se ao paradigma de execução sequencial
- Processamento das diferentes posições de memória em paralelo
- Memória deve ser implementada como mais de um módulo



MISD (Multiple Instruction Single Data)

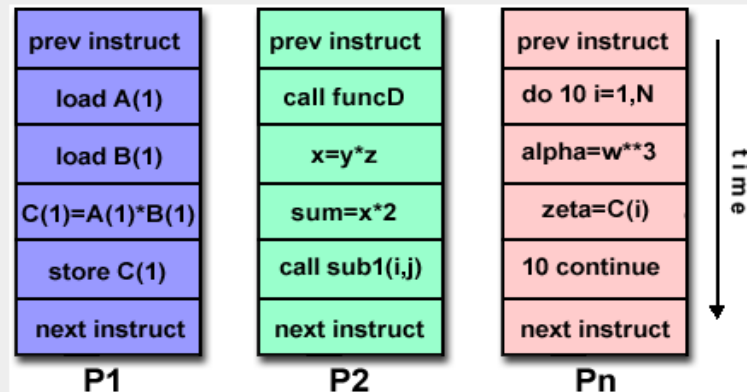
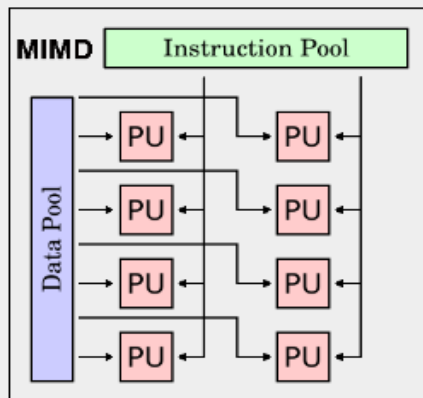
- Múltiplas unidades de processamento, com sua unidade de controle, recebendo fluxo diferente de instruções
- Os processadores executam suas diferentes instruções sobre mesmo fluxo de dados
- Diferentes instruções operam mesma posição de memória ao mesmo tempo, executando instruções diferentes



Além de ser tecnicamente impraticável, essa classe é considerada vazia

MIMD (Multiple Instruction Multiple Data)

- Cada controlador recebe 1 fluxo de instruções e repassa para seu processador para que seja executado sobre seu fluxo de instruções
- Cada processador executa seu próprio programa sobre seus próprios dados de forma assíncrona
- Princípio MIMD é bastante genérico
- Qualquer grupo de máquinas, se analisado como uma unidade pode ser considerado uma máquina MIMD
- Igualmente ao SIMD, unidade de memória não pode ser implementada como um único módulo



Plataforma de Execução Paralela

► Diferentes plataformas do **MIMD** de acordo com os seguintes critérios

- espaço de endereçamento
- mecanismo de comunicação

► Podem ser agrupadas em quatro grupos

SMPs (*Symmetric MultiProcessors*)

MPPs (*Massively Parallel Processors*)

Cluster ou **NOWs** (*Network Of Workstations*)

Grades Computacionais

SMPs (*Symmetric MultiProcessors*)

- ▶ SMPs ou Multiprocessadores

 - único espaço de endereçamento lógico

- ▶ mecanismo de hardware (memória centralizada)

- ▶ comunicação ⇒ espaço de endereçamento compartilhado

- ▶ operações de *loads* e *stores*

- ▶ Acesso a memória é realizada através de leitura (load) e escrita (store), caracterizando desta forma, a comunicação entre processadores

SMPs (*Symmetric MultiProcessors*)

- ▶ Sistema homogêneo
- ▶ Compartilhamento
- ▶ Compartilhamento total da mesma memória
- ▶ Uma única cópia do Sistema Operacional
- ▶ Imagem única do sistema
- ▶ Excelente conectividade
- ▶ fortemente acoplados
- ▶ Não escalável
- ▶ Exemplos:
 - ▶ Sun HPC 10000 (StarFire), SGI Altix, SGI Origin, IBM pSeries, Compac AlphaServer

MPPs (*Massively Parallel Processors*)

- ▶ Diferem quanto a implementação física
- ▶ Módulos ou elementos de processamento contendo:
 - ▶ múltiplos processadores com memória privativa
 - ▶ computadores completos
- ▶ Espaço de endereçamento
 - ▶ não compartilhado - **memória distribuída**
- ▶ Comunicação
 - ▶ troca de mensagens
- ▶ Rede de interconexão
 - ▶ diferentes topologias
- ▶ Fracamente acoplados
- ▶ Escaláveis

MPPs (*Massively Parallel Processors*)

- ▶ Sistema homogêneo ou heterogêneo
- ▶ Interconexão: redes dedicadas e rápidas
- ▶ Cada nó executa sua própria cópia do Sistema Operacional
- ▶ Imagem única do sistema
- ▶ visibilidade dos mesmos sistemas de arquivo
- ▶ Um escalonador de tarefas
- ▶ partições diferentes para aplicações diferentes

MPPs (*Massively Parallel Processors*)

- ▶ Partições dedicadas a cada aplicação
- ▶ Aplicações não compartilham recursos
- ▶ Pode ocorrer que uma aplicação permaneça em estado de espera
- ▶ Exemplos:
 - ▶ Cray T3E, IBM SP2s, *clusters* montados pelo próprio usuário, com propósito de ser um MPP

Cluster de computadores ou NOWs

- ▶ “Cluster é um sistema distribuído que consiste na coleção de computadores interconectados, usados como um sistema único.” (Gregory F. Pfister).
- ▶ Sistema de processamento de dados **paralelo** e/ou **distribuído**.
- ▶ **Agregar** computadores pessoais.
- ▶ Interconexão: **redes locais**
- ▶ Computadores conectados de forma **cooperativa**.
- ▶ Nós: elementos de processamento = processador + memória
- ▶ Diferenças em relação a MPPs:
 - ▶ não existe um escalonador centralizado
 - ▶ redes de interconexão tendem a ser mais lentas

Cluster de computadores ou NOWs

- ▶ Resultado das diferenças:
- ▶ Cada nó tem seu próprio escalonador local
- ▶ Compartilhamento de recursos ⇒ sem partição dedicada a uma aplicação
- ▶ Aplicação ⇒ deve considerar impacto no desempenho
 - ⇒ não tem o sistema dedicado
- ▶ Possibilidade de compor um sistema de alto desempenho e um baixo custo (principalmente quando comparados com MPPs).

Cluster de computadores ou NOWs

- ▶ Utilização de computadores
 - ▶ independentes
 - ▶ geograficamente distantes
- ▶ Diferenças: *clusters* X *grades*
- ▶ heterogeneidade de recursos
- ▶ alta dispersão geográfica (escala mundial)
- ▶ compartilhamento
- ▶ múltiplos domínios administrativos
- ▶ controle totalmente distribuído

Computação em Cluster

- Um conjunto de computadores (PCs)
- não necessariamente iguais ✉ heterogeneidade
- Filosofia de imagem única
- Conectadas por uma rede local

Para atingir tais objetivos, necessidade de uma camada de software ou *middleware*

Grades Computacionais

- ▶ Componentes
 - ▶ PCs, SMPs, MPPs, *clusters*
 - ▶ controlados por diferentes entidades ⇒ diversos domínios administrativos
- ▶ Não têm uma imagem única do sistema a princípio
- ▶ Vários projetos tem proposto o desenvolvimento de middlewares de gerenciamento ⇒ camada entre a infra-estrutura e as aplicações a serem executadas na grade computacional
- ▶ Aplicação deve estar preparada para:
 - ▶ Dinamismo
 - ▶ Variedade de plataformas
 - ▶ Tolerar falhas

Grades Computacionais

- ▶ Sistema não dedicado e diferentes plataformas
- ▶ Usuários da grades devem obter autorização e certificação para acesso aos recursos disponíveis na grade computacional
- ▶ Falhas nos recursos tanto de processamento como comunicação são mais freqüentes que as outras plataformas paralelas
- ▶ Mecanismos de tolerância a falhas devem tornar essas flutuações do ambiente transparente ao usuário
- ▶ Para utilização eficiente da grade computacional
- ▶ Gerenciamento da execução da aplicação através de políticas de escalonamento da aplicação ou balanceamento de carga
- ▶ Escalonamento durante a execução da aplicação se faz necessário devido as variações de carga dos recursos da grade

Computação em Grid

- Computação em Cluster foi estendido para computação ao longo dos sites distribuídos geograficamente conectados por redes metropolitanas

Grid Computing

- Heterogêneos
- Compartilhados
- Aspectos que devem ser tratados
- Segurança
- Falhas de recursos
- Gerenciamento da execução de várias aplicações


Distribuição de Memória

Refere-se à localização física da memória

- Memória distribuída (*distributed memory*)

- Memória implementada com vários módulos
- Cada módulo fica próximo de um processador

- Memória centralizada (*centralized memory*)

- Memória encontra-se à mesma distância de todos os processadores 
- Independentemente de ter sido implementada com um ou vários módulos

Compartilhamento de Memória

Refere-se ao espaço de endereçamento dos processadores

- Memória compartilhada (*shared memory*)

- Único espaço de endereçamento usado para comunicação entre processadores

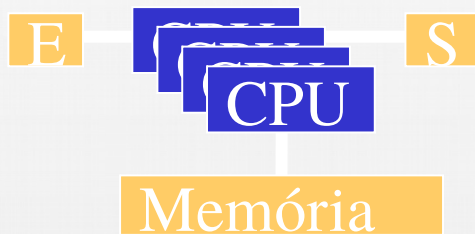
- Operações de *load* e *store*

- Memória não compartilhada

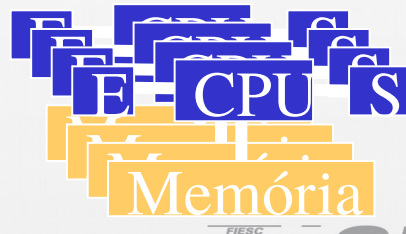
- Múltiplos espaços de endereçamento privados (*multiple private address spaces*) ✉ um para cada processador

- Comunicação através de troca de mensagens ✉ operações **send** e **receive**

- Conforme compartilhamento de memória as máquinas podem ser



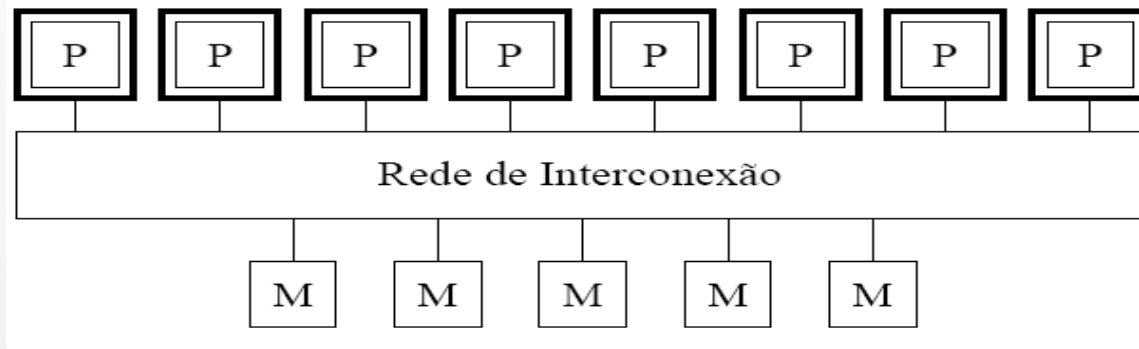
Multiprocessadores



Multicomputadores

Multiprocessadores

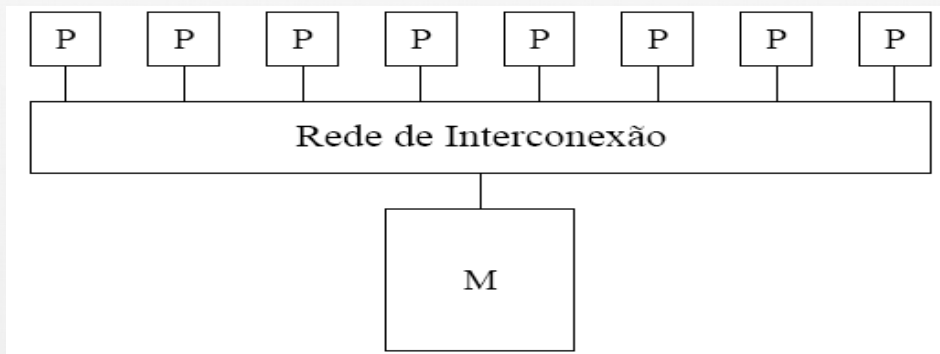
- Máquina paralela construída a partir da replicação de processadores de uma arquitetura convencional
- Todos processadores (P) acessam memórias compartilhadas (M) através de uma infra-estrutura de comunicação
- Possui apenas um espaço de endereçamento
- Comunicação entre processos de forma bastante eficiente (*load e store*)



- Em relação ao tipo de acesso às memórias do sistema, multiprocessadores podem ser classificados como:
 - UMA
 - NUMA, NCC-NUMA, CC-NUMA
 - COMA

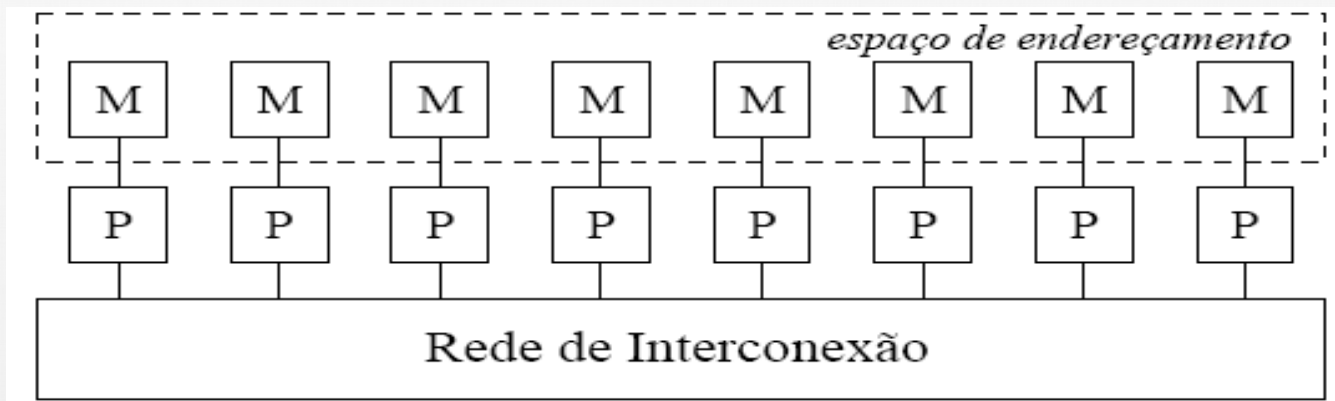
Acesso Uniforme à Memória (Uniform Memory Access - UMA)

- Memória centralizada
 - Encontra-se à mesma distância de todos processadores
- Latência de acesso à memória
 - Igual para todos processadores
- Infra-estrutura de comunicação
 - Barramento é a mais usada ☒ suporta apenas uma transação por vez
 - Outras infra-estruturas também se enquadram nesta categoria, se mantiverem uniforme o tempo de acesso à memória



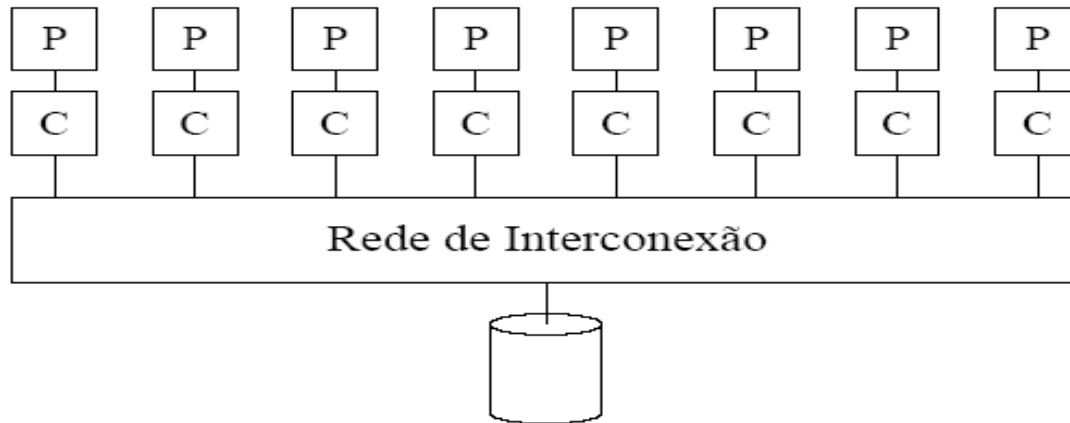
Acesso Não Uniforme à Memória (Non-Uniform Memory Access - NUMA)

- Memória Distribuída
- Espaço de endereçamento único
- Memória implementada com múltiplos módulos associados a cada processador
- Comunicação processador-memórias não locais através da infra-estrutura de comunicação
- Tempo de acesso à memória local < tempo de acesso às demais ✉ Acesso não uniforme ✉
Distância das memórias variável ✉ depende do endereço



Arquiteturas de Memória Somente com Cache (Cache-Only Memory Architecture - COMA)

- Memórias locais estão estruturadas como memórias *cache*
- São chamadas de COMA *caches*
- COMA *caches* têm muito mais capacidade que uma *cache* tradicional
- Arquiteturas COMA têm suporte de hardware para a replicação efetiva do mesmo bloco de *cache* em múltiplos nós
- Reduz tempo global para pegar informações

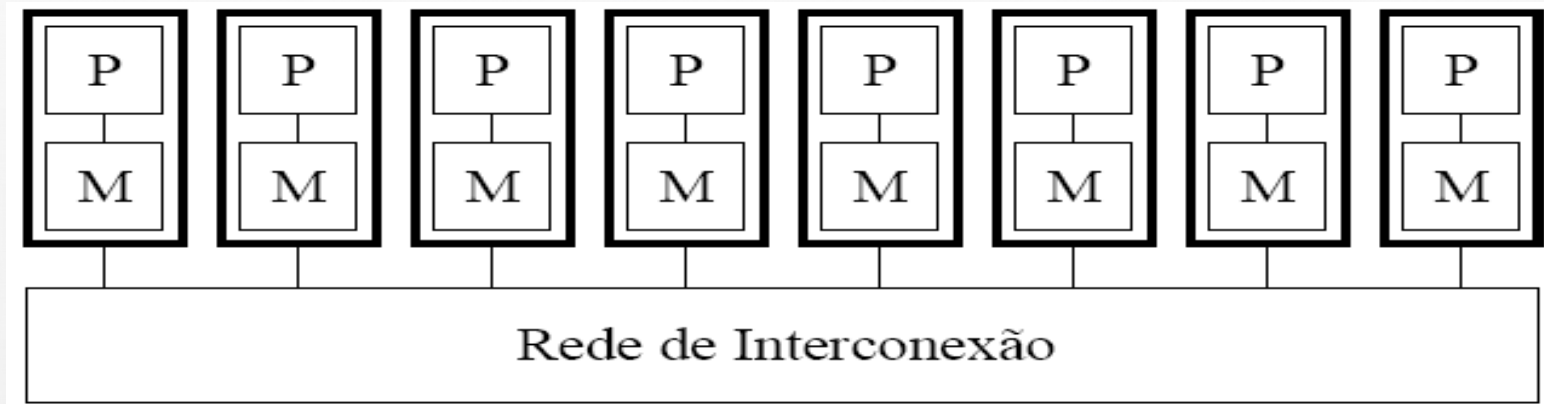


Multicomputadores

- Processadores (P) possuem memória local (M) de acesso restrito
- Memórias de outros processadores são remotas ✉ Possuem espaços de endereçamento distintos
- Não é possível uso de variáveis compartilhadas
 - Troca de informações feita por **envio de mensagens**
 - Máquinas também chamadas *message passing systems*
- **Multicomputadores** são construídos a partir da replicação de toda arquitetura convencional, não apenas do processador, como nos **multiprocessadores**

Sem Acesso à Memória Remota (Non-Remote Memory Access - NORMA)

- Multicomputadores são classificados como NORMA
- Replicação inteira da arquitetura faz que cada nó só consiga endereçar sua memória local



Visão Geral da Classificação Segundo Modelo de Memória

