Universidade Federal do Rio de Janeiro
Pós-Graduação em Informática
DCC/IM - NCE/UFRJ

Arquiteturas de Sistemas de Processamento Paralelo

Arquiteturas MIMD

Gabriel P. Silva

Arquiteturas MIMD com Memória Distribuída

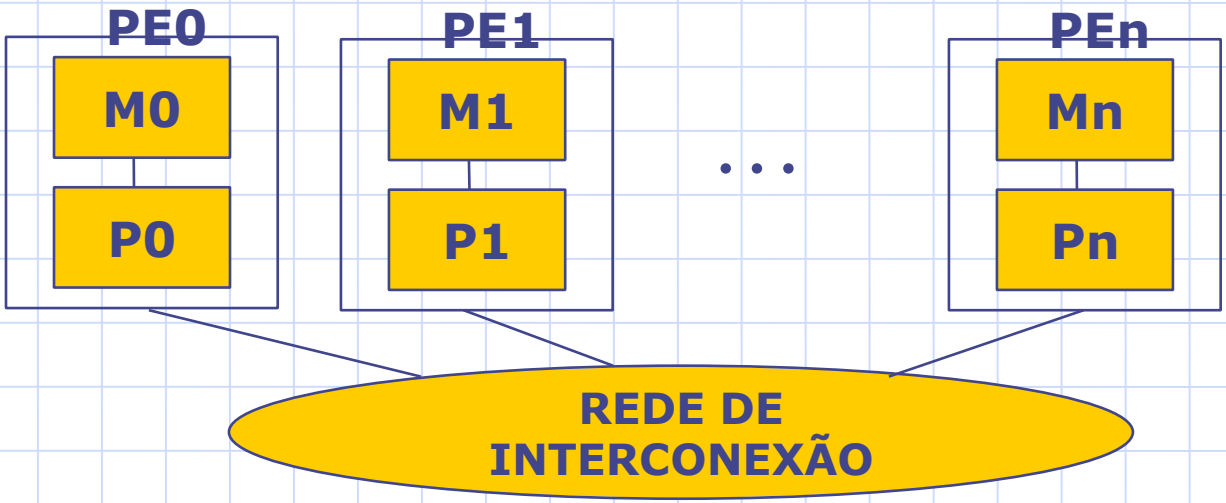
MIMD com Memória Distribuída

- Cada processador enxerga apenas o seu espaço de memória.
- Vantagens:
  - Altamente escalável e permitem a construção de processadores maciçamente paralelos.
  - A troca de mensagens resolve tanto o problema da comunicação como o da sincronização.

MIMD com Memória Distribuída

- **Desvantagens:**
  - **Necessidade de fazer uma boa distribuição de carga entre os processadores, quer seja automaticamente, quer seja manualmente.**
  - **É necessário evitar as situações de "deadlock", tanto no nível de aplicação como no nível do sistema operacional.**
  - **Modelo de programação menos natural.**

MIMD com Memória Distribuída



MIMD com Memória Distribuída

- Os processos e suas áreas de dados estão localizados na memória local
- Processos executando em diferentes elementos processadores devem enviar mensagens através da rede de interconexão para se comunicar.
- Quando são necessários acessos aos dados remotos, esses são solicitados através do envio de mensagens para o processo que os controla
- Enquanto aguarda a resposta, dois procedimentos são possíveis:
  - Suspender o processo solicitante e ativar outro processo
  - O processo pode continuar a fazer computação útil até que precise da resposta, quando então aguarda em "loop" de espera

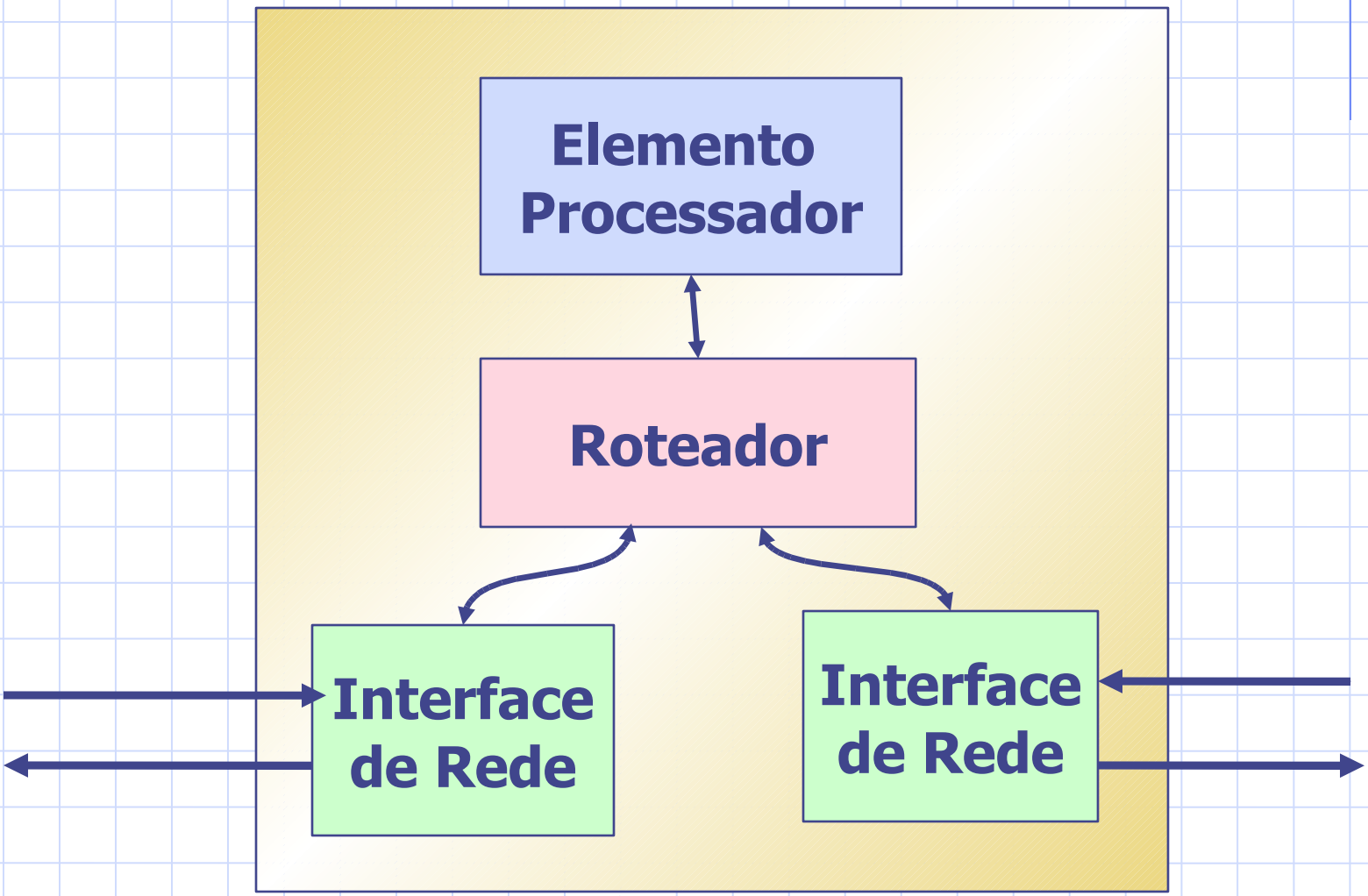
MIMD com Memória Distribuída

- Para a redução da comunicação entre processadores, dois fatores devem ser considerados:
  - Como melhor particionar o programa paralelo em processos.
  - Como mapear os processos nos processadores de modo a diminuir a comunicação
- Esses problemas são de difícil resolução (NP completo) e ainda podem implicar em problemas de balanceamento de carga
- Isto levou à adoção dos sistemas com memória distribuída compartilhada, onde essas questões são resolvidas pelo sistema de hardware/software subjacente.

MIMD com Memória Distribuída

- Ainda assim as arquiteturas por troca de mensagem tem desempenho de pico para aplicações com grande quantidade de paralelismo.
- Comunicação entre os nós é realizada através de conexões diretas denominadas links ou canais.
- Os nós são compostos por três elementos principais:
  - Processador de Computação+Memória local (EP)
  - Processador de Comunicação
  - Roteador ou Unidade de Chaveamento
- Os nós são interligados por uma rede de interconexão estática.

MIMD com Memória Distribuída



Arquiteturas MIMD com Memória Compartilhada

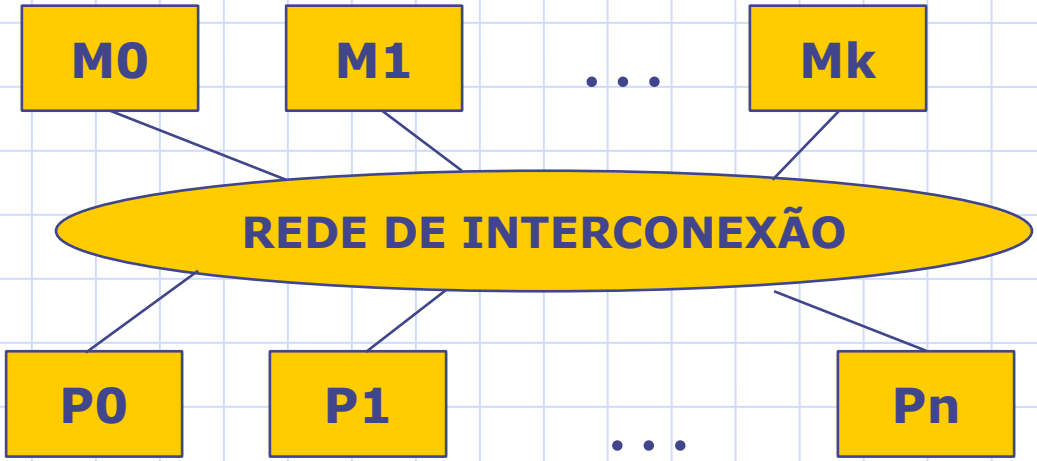
MIMD com Memória Compartilhada

- Cada processador consegue ter acesso a todo o espaço de memória da arquitetura
- Vantagens:
  - Não há necessidade de particionar o código ou dados, logo técnicas de programação para uniprocessadores podem ser facilmente adaptados para ambientes multiprocessadores.
  - Não há necessidade de movimentar fisicamente os dados quando dois ou mais processadores se comunicam. Como resultado a comunicação entre processos é bastante eficiente.

MIMD com Memória Compartilhada

- **Desvantagens:**
  - **Há necessidade do uso de primitivas especiais de sincronização quando do acesso a regiões compartilhadas na memória.**
  - **Falta de escalabilidade devido ao problema de contenção de memória. Depois de um determinado número de processadores a adição de mais processadores não aumenta o desempenho.**

MIMD com Memória Compartilhada



Análise de Escalabilidade em Memória Compartilhada

- **Problemas de Escalabilidade**
  - Tolerar e esconder as latências de acesso a dados remotos.
  - Tolerar e esconder o tempo de espera devido a sincronização entre processos paralelos.
- **Soluções:**
  - O uso de uma rede de interconexão com alto "throughput" e baixa latência melhora a escalabilidade.
  - Uso de memórias "caches" locais reduz o problema da contenção, mas exige o uso de algoritmos para manutenção da coerência de dados.

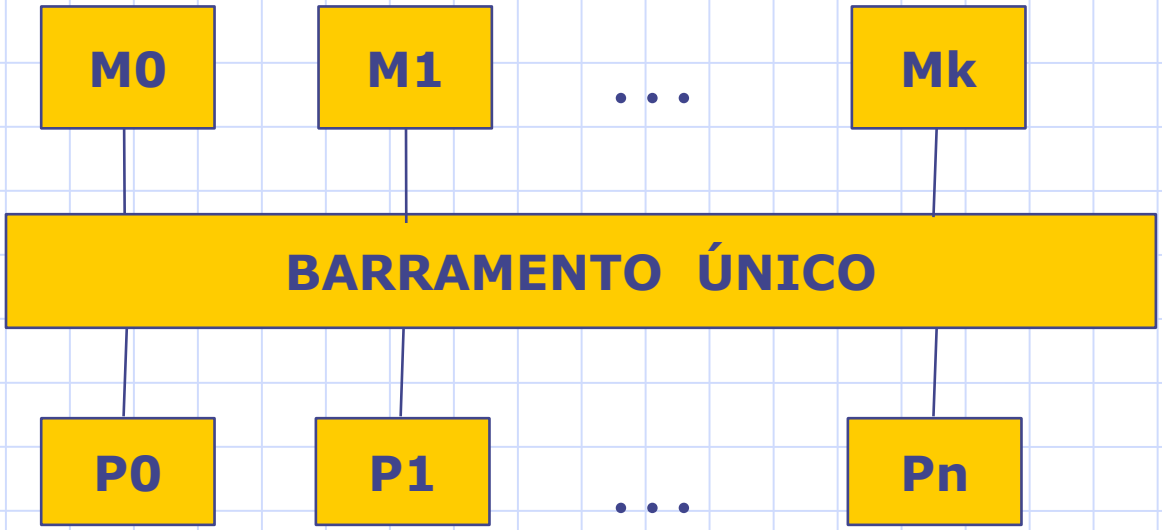
Análise de Escalabilidade em Memória Compartilhada

- **Soluções:**
  - Busca avançada dos dados/páginas.
  - Uso de "threads" e um mecanismo rápido de troca de contexto entre "threads".
  - A memória logicamente compartilhada pode ser implementada com o uso de um conjunto de memórias locais. Este esquema é denominado "Arquitetura de Memória Compartilhada Distribuída", que pode ser dividido em três classes:
    - NUMA (Non-uniform Memory Access)
    - CC-NUMA (Cache Coherent Non-uniform Memory Access)
    - COMA (Cache-Only Memory Access)

Arquiteturas UMA

- Arquiteturas com memória única global.
- Tempo de acesso uniforme para todos os nós de processamento.
- Nós de processamento e memória interconectados através de barramento único.
- Número reduzido de nós de processamento.
- Coerência de cache mantida por "hardware" com o uso da técnica de "snooping".

Arquiteturas UMA



Barramentos

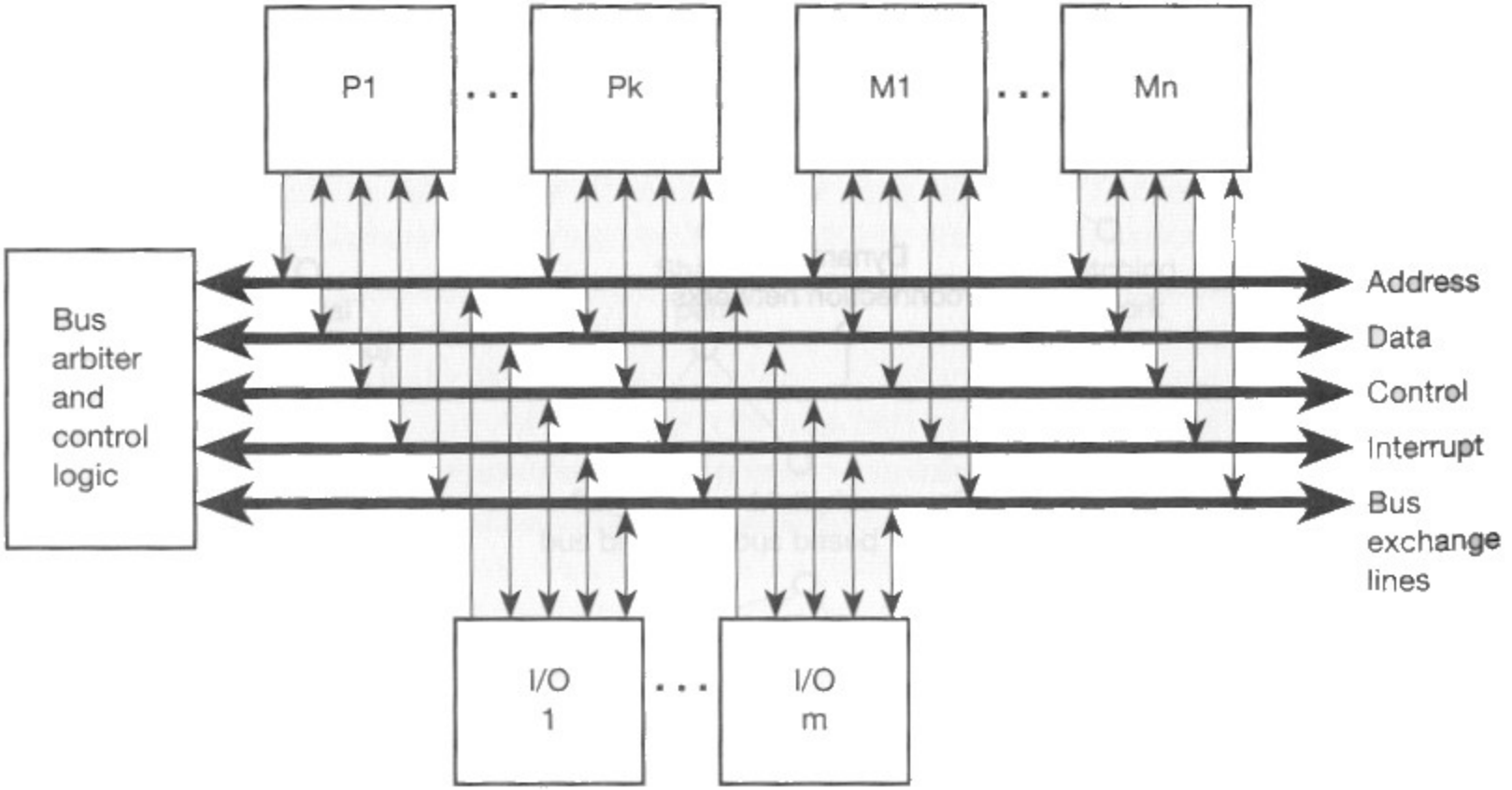
Barramento Único Compartilhado

- **Vantagens:**
  - Organização simples
  - Baixo custo
- **Desvantagens:**
  - A contenção aumenta significativamente com o número de processadores
- **Técnicas para a diminuição da contenção:**
  - Memória privativa
  - Memória cache com coerência
  - Múltiplos barramentos
- **O uso de memória privativa e memória cache aumenta o número máximo de processadores de 3-5 para cerca de 30**

Barramento Único Compartilhado

- O uso de múltiplos processadores requer modificações no acesso à memória:
  - Os acessos de escrita são divididos em duas fases:
    - Envio dos dados e endereço
    - Escrita dos dados na memória
  - Os acessos de leitura são divididos em três fases:
    - Envio do endereço
    - Leitura dos dados da memória
    - Envio dos dados para o processador
  - Um árbitro para ordenar o acesso dos diversos processadores deve ser instalado

Barramento Único Compartilhado



Múltiplos Barramentos Compartilhados

- Podem ser de diversos tipos:
  - Unidimensionais
  - Bi ou tri-dimensionais
  - Cluster
  - Hierárquico
- Os dois primeiros tipos são classificados como arquiteturas UMA. Os dois últimos, como arquiteturas NUMA, pois nesse caso, o tempo de acesso a um módulo de memória dentro de um "cluster" ou de uma mesma hierarquia é diferente do tempo de acesso a um módulo remoto.

Múltiplos Barramentos Compartilhados

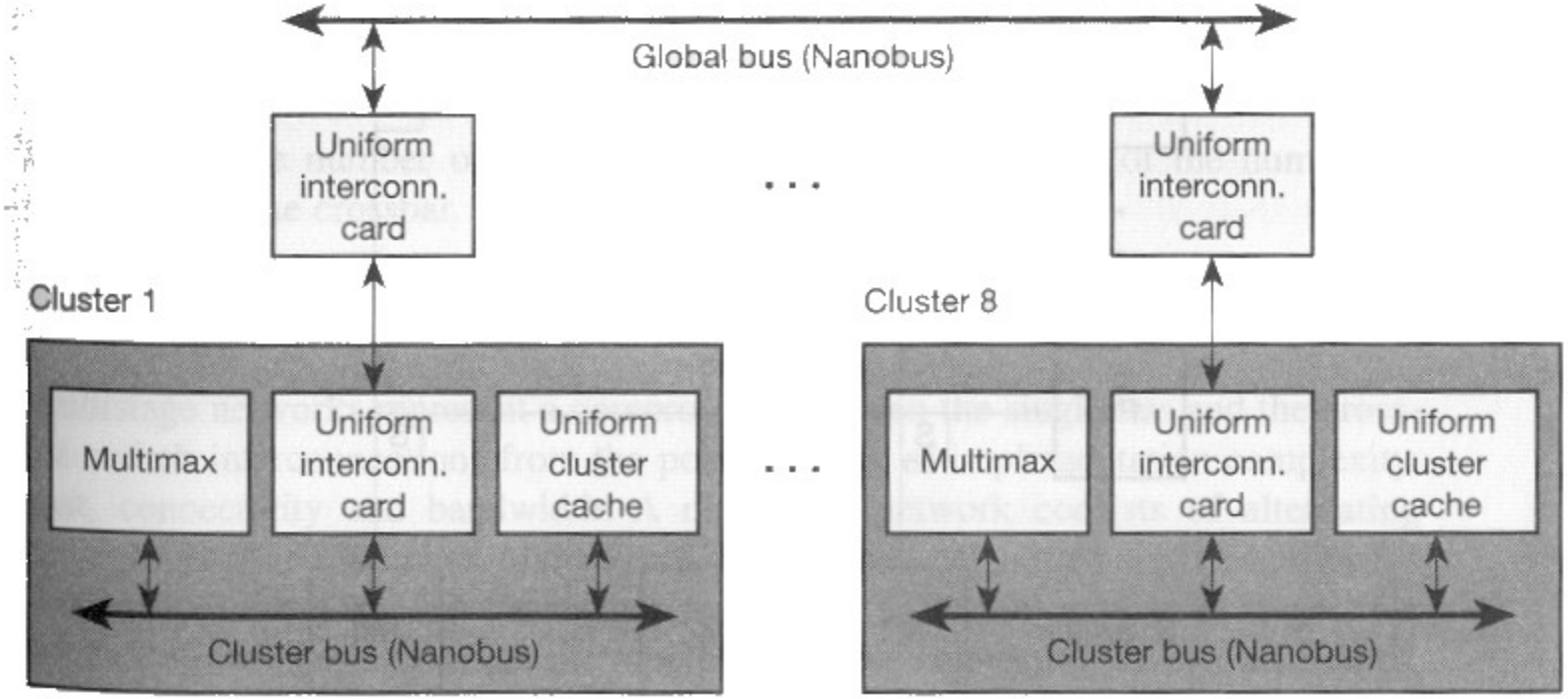
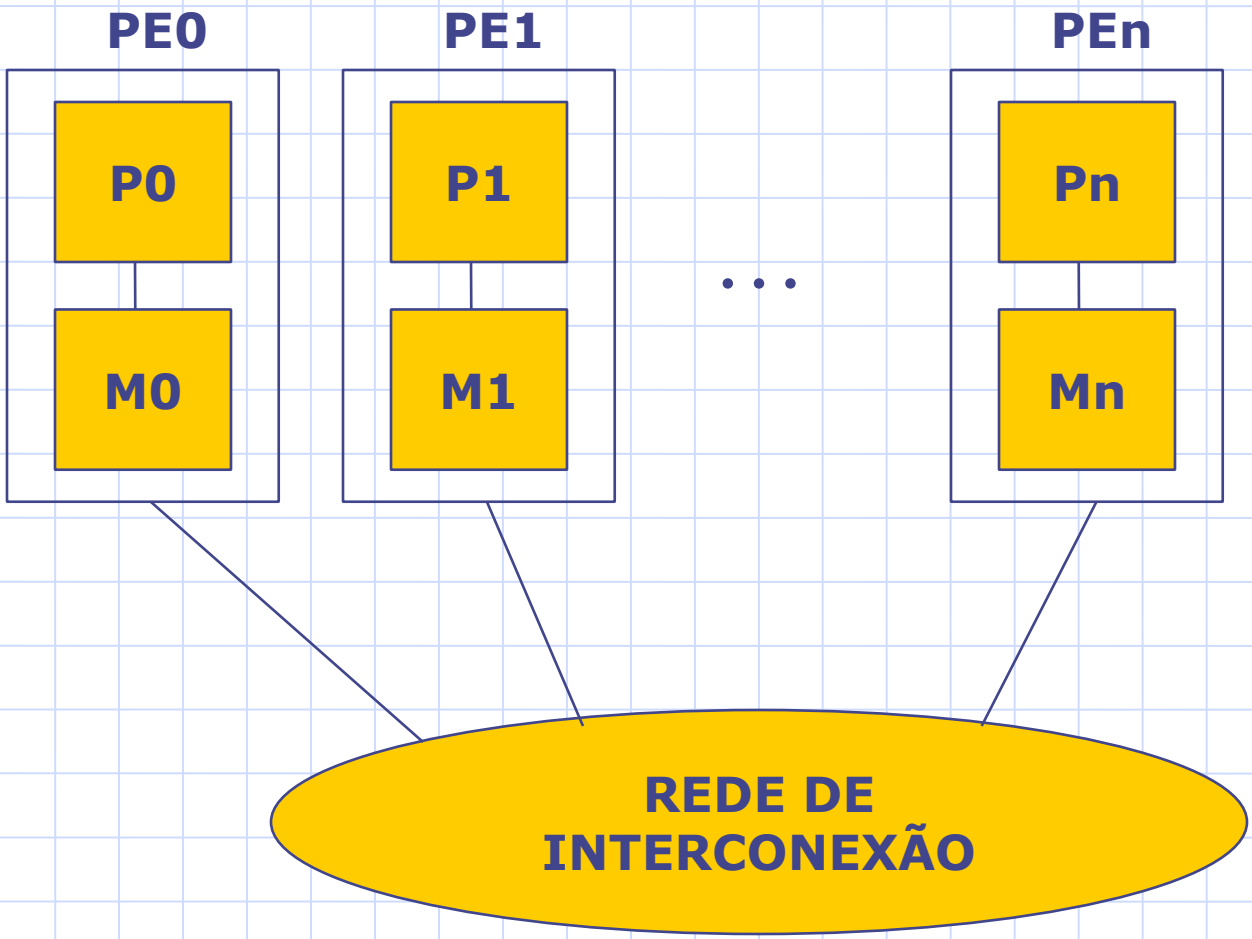


Figure 18.12 The GigaMax cluster architecture.

Arquiteturas NUMA

- Nessas arquiteturas a memória é dividida em tantos blocos quanto forem os processadores do sistema, e cada bloco de memória é conectado via barramento a um processador como memória local.
- O acesso aos dados que estão na memória local é muito mais rápido que o acesso aos dados em blocos de memória remotos.
- Esta diferença faz com que sejam necessários cuidados especiais ao se programar em arquiteturas deste tipo.
- Apesar do uso de memória compartilhada, as arquiteturas NUMA mais modernas oferecem bibliotecas para programação utilizando troca de mensagens.
- Exemplo: Cray T3D

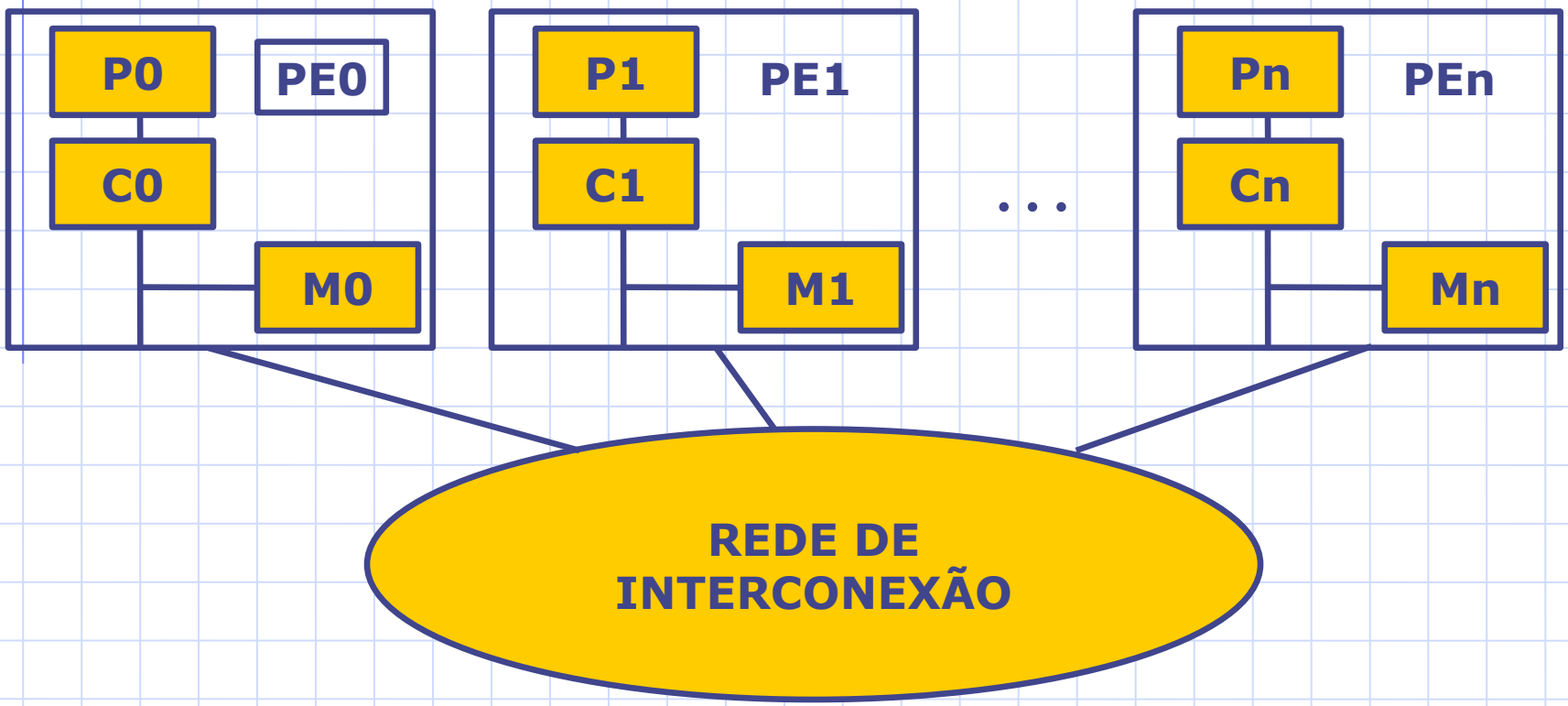
Estrutura da Arquitetura NUMA



Arquiteturas CC-NUMA

- Solução de compromisso entre as arquiteturas NUMA e COMA.
- Cada nó processador possui uma cache local para reduzir o tráfego na rede de interconexão.
- O balanceamento de carga é realizado dinamicamente pelos protocolos de coerência das caches.
- Exemplo: Convex SPP1000, Stanford DASH e MIT Alewife

Arquiteturas CC-NUMA

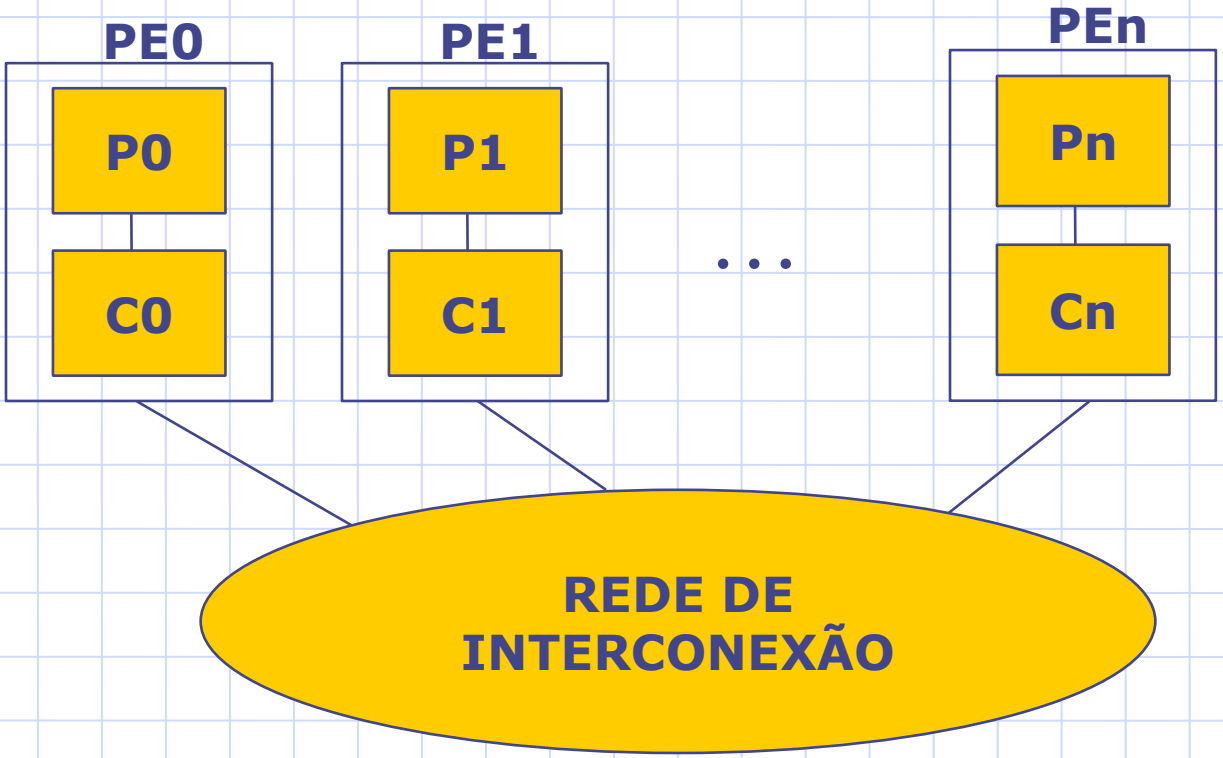


Problemas de Escalabilidade

- Tolerar e esconder as latências de acesso a dados remotos.
- Tolerar e esconder o tempo de espera devido a sincronização entre processos paralelos.
- Soluções:
  - Uso de memória cache
  - Pré-busca
  - Uso de “threads” e um mecanismo rápido de troca de contexto entre “threads”

Arquiteturas COMA

Arquiteturas COMA



Arquiteturas COMA

- Assemelham-se a uma arquitetura NUMA, onde cada nó de processamento possui uma parte da memória global.
- O particionamento dos dados entre as memórias de cada nó não é estático → as memórias funcionam como caches de nível 3.
- O problema de partição de dados e balanceamento dinâmico de carga é realizado automaticamente.
- Conforme o algoritmo de coerência utilizado, os dados migram automaticamente para as caches locais dos processadores onde é mais necessária.
- Exemplo: KSR-1 e DDM

Cache-Only Memory Architectures

- Quando ocorre um "miss" na AM, o bloco de cache é copiado de um nó remoto que possui cópia do bloco.
- Quando ocorre uma substituição de bloco na AM, o sistema deve garantir que pelo menos uma cópia do bloco substituído continue existindo no sistema. Caso o bloco substituído seja o único existente, ele deve ser relocado, possivelmente em outra AM.
- Possuem um "overhead" de memória, pois parte da memória das AM's não é alocada para as aplicações ou S.O. de modo a facilitar a replicação e migração dos dados.

Arquiteturas COMA Hierárquicas

- As arquiteturas COMA podem ser de 4 tipos: hierárquica, FLAT, S-COMA e MS-COMA.
- As arquiteturas hierárquicas são organizadas em uma árvore de hierarquia.
- Os processadores são conectados às folhas dessa árvore
- Cada nível de hierarquia inclui um diretório com informação de "status" dos blocos de memória existentes desde as folhas até aquele nível de hierarquia.
- Para encontrar um bloco, o processador emite um pedido que sobe gradativamente na hieraquia até atingir um nível cuja sub-árvore contenha o bloco desejado.

Arquiteturas FLAT COMA

- O diretório que armazena a informação sobre cada bloco de memória tem uma localização fixa, no chamado "home node" do bloco. Os blocos podem migrar livremente, mas os seus diretórios não.
- Quando ocorre uma falha na busca de um bloco em uma AM, a solicitação pelo bloco é direcionada para o seu "home node". De lá, a solicitação é redirecionada para o nó que contém uma cópia do bloco desejado.
- O "home node" pode não conter uma cópia de um bloco, mesmo que esse bloco não tenha sido escrito. No entanto, ele sempre sabe em que nó há uma cópia do bloco.
- As arquiteturas FLAT COMA tendem a produzir latências bem menores para a obtenção de uma cópia de um bloco em caso de falha na AM do que as arquiteturas hierárquicas.

S-COMA - Arquitetura

- **Tipicamente construída com nós SMP conectados através de uma rede de interconexão de alto desempenho.**
- **Cada nó roda sua própria cópia do SO com mínimas extensões para a camada de memória virtual para permitir que dados sejam compartilhados entre diferentes nós.**
- **Cada nó possui um Adaptador de Memória Compartilhada (SMA) na qual permite que se observe cada acesso à memória pelo processador.**
- **Deve conter uma interface de rede integrada que permite a entrada e saída de mensagens de coerência no nó.**

Arquiteturas S-COMA

- O modelo S-COMA (Simple COMA) tenta reduzir a complexidade do projeto, transferindo parte dessa complexidade para o software.
- Nesse modelo, o S.O. aloca espaço com granularidade de uma página na AM, para a chegada de novos blocos. A MMU de cada nó possui mapeamentos para páginas locais.
- Quando ocorre um "page fault", o S.O. aloca um "page frame" na AM local e, em seguida, o "hardware" localiza uma cópia do bloco desejado e a insere na página recém-alocada. O resto da página fica marcada como inválida.

S-COMA Gerenciamento da AM pelo SO

- O gerenciamento da AM por hardware em uma arquitetura COMA é a parte mais custosa em hardware. S-COMA soluciona este problema fazendo com que o SO seja responsável pela gerência desta parte complexa.
- A alocação e substituição de páginas é feita então pelo SO.

Arquiteturas S-COMA

- Quando um bloco inválido é acessado, o hardware detecta isso e, de modo totalmente transparente, busca uma cópia do bloco desejado em outro nó.
- O endereço físico de um dado bloco na AM é definido de forma independente por cada MMU de cada nó. Logo, um mesmo bloco compartilhado pode ter endereços físicos distintos em cada nó. Portanto, cada nó tem uma tabela que traduz os endereços físicos em identificadores globais e vice-versa.
- Um exemplo de arquitetura é a "Illinois Aggressive Coma Multiprocessor Project".

MS-COMA

- A arquitetura MS-COMA (Multiplexed Simple COMA) tenta reduzir a fragmentação de memória que pode ocorrer nas arquiteturas S-COMA.
- Na arquitetura MS-COMA, múltiplas páginas virtuais em um dado nó podem mapear numa mesma página física simultaneamente.
- Uma única página física pode, portanto, conter blocos pertencentes a diferentes páginas virtuais.
- Dois blocos como mesmo "offset" em relação às páginas virtuais, que mapeiam em uma mesma página física, não podem residir conjuntamente na AM.

S-COMA e R-NUMA

- S-COMA (Simple COMA) é um modelo que tenta reduzir a complexidade imposta na arquitetura COMA transferindo parte desta complexidade para o software.
- R-NUMA (Reactive NUMA) são arquiteturas que suportam tanto o modelo S-COMA como o modelo CC-NUMA (Cache Coherent NUMA). A escolha se dará segundo critérios estabelecidos como forma de aumentar o desempenho.

R-NUMA

- As arquiteuras Reactive NUMA suportam tanto o modelo NUMA-RC quanto o modelo S-COMA em uma base página a página.
- O sistema operacional usa a informação sobre o número de "re-buscas" realizadas por um nó para blocos de uma determinada página para mudar para o modo S-COMA.
- Um "re-busca" é um acesso que traz um bloco que foi retirado da memória local, devido a um "overflow", de volta para a memória local.
- Um contador de "hardware" verifica o número de "re-buscas" de um nó para um página. Quando o contador atinge um determinado limite, o S.O muda a página para o modo S-COMA para aquele nó.
- Os multiprocessadores WildFire da Sun e o PRISM da IBM utilizam esse princípio.

R-NUMA

- Reactive-NUMA (R-NUMA) é uma arquitetura híbrida que combina parte S-COMA e CC-NUMA sendo projetada no topo de uma CC-NUMA.
- Há um único espaço de endereçamento físico em toda a máquina e os bits mais significativos do endereço codificam o identificador do nó processador.
- Páginas acessadas no modo CC-NUMA utilizam este endereço físico mesmo que o endereço esteja em um nó remoto.
- Páginas acessadas no modo S-COMA são replicadas na memória local e o endereço físico local é usado.

R-NUMA

- A SMA tem então a capacidade de trabalhar com transações de memória tanto no modo S-COMA como no modo CC-NUMA.
- Numa transação feita no barramento o SMA checa o endereço físico. Caso o identificador de nó (no endereço físico) for fora daquele nó, é um acesso claro a um bloco no modo CC-NUMA.
- Se o endereço físico for referente ao próprio nó, o SMA verifica se é um acesso ao modo S-COMA ou CC-NUMA e o protocolo de coerência deverá se adaptar a uma das duas opções.

R-NUMA

- Para cada bloco existe um contador que indica o número de re-buscas. Os contadores são iniciados com 0. Quando chegam a um valor determinado, o nó é interrompido para remapear a página no modo S-COMA.
- Se a página for substituída, o contador vai a zero e ela é remapeada como CC-NUMA.

Desempenho

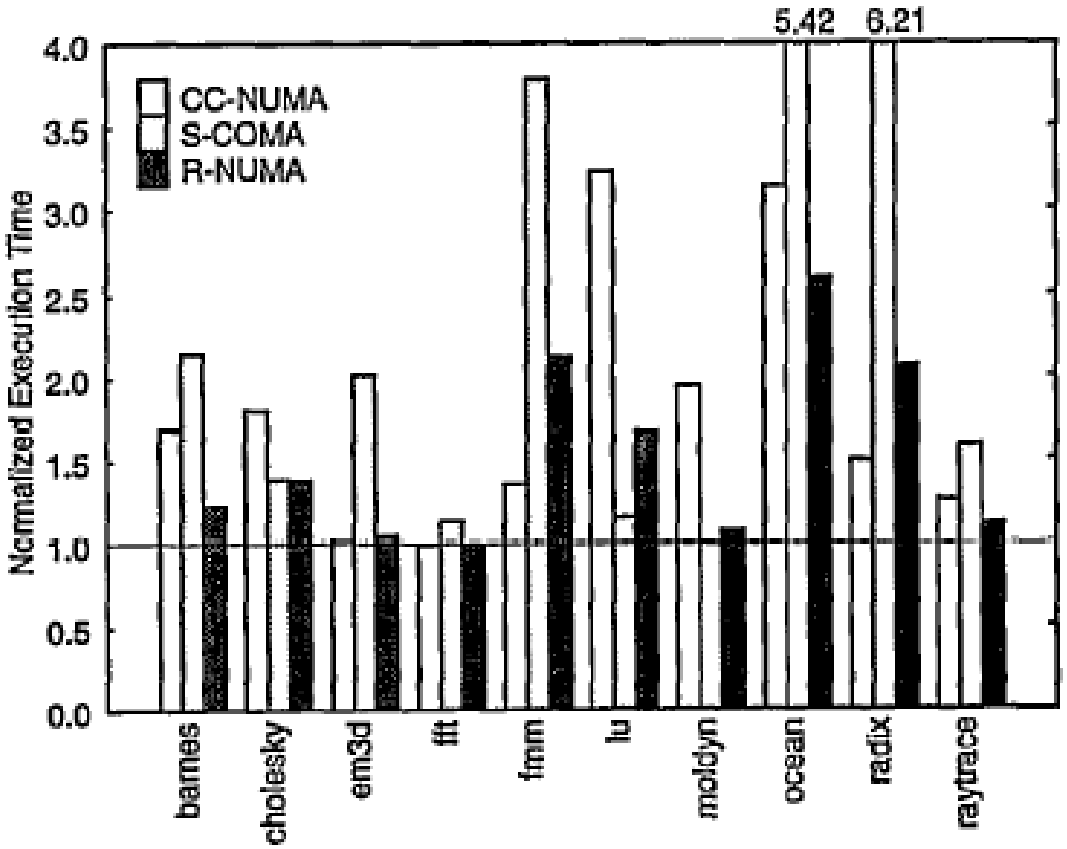


FIGURE 6. Comparing performance of CC-NUMA, S-COMA and R-NUMA.

The figure plots execution times on a CC-NUMA with a 32-Kbyte block cache, S-COMA with 320-Kbyte page cache, and R-NUMA with a 128-byte block cache, a 320-Kbyte page cache, and a relocation threshold value of 64. The numbers are normalized to a CC-NUMA with an infinite block cache, i.e., a block cache that can hold all of the referenced remote data.