

Arquitetura de Computadores Paralelos

Computação Paralela

Faculdade de Computação e Informática

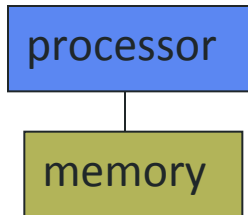
Outline

- ❑ Tipos de arquiteturas paralelas
- ❑ Paralelismo em nível de instrução
- ❑ Processamento vetorial
- ❑ SIMD
- ❑ Memória compartilhada
 - Organização de Memória: UMA, NUMA
- ❑ Interconnection networks
- ❑ Clusters

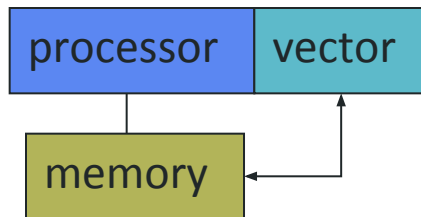
Tipos de Arquitetura Paralelas

- Uniprocessor

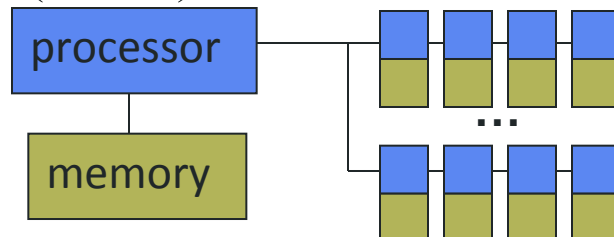
- Scalar processor



- Vector processor



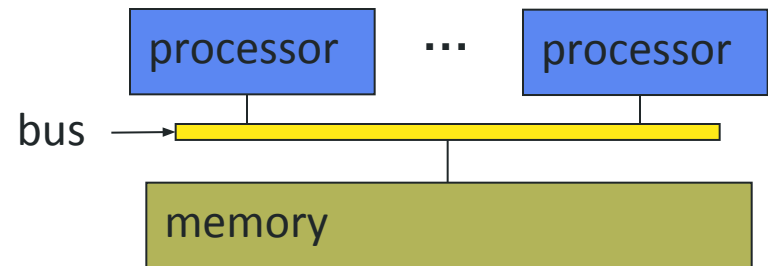
- Single Instruction Multiple Data (SIMD)



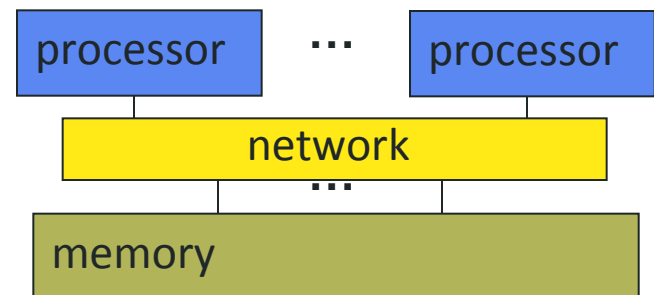
- Shared Memory

Multiprocessor (SMP)

- Shared memory address space
- Bus-based memory system



- Interconnection network

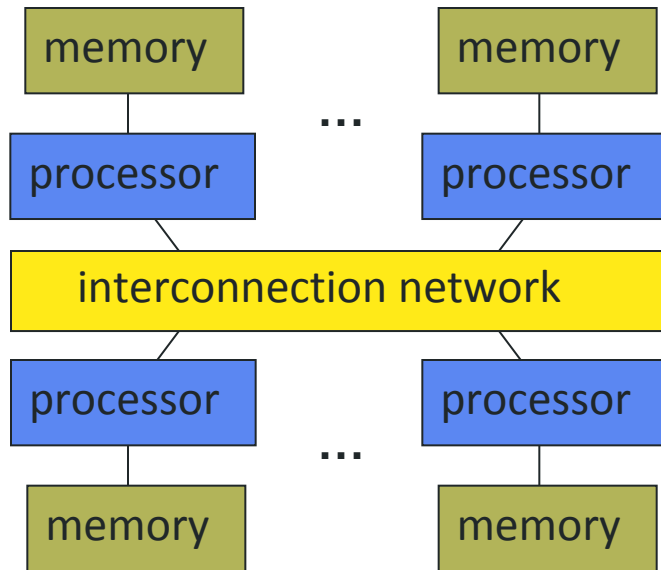


Tipos de Arquitetura Paralelas - (2)

- Distributed Memory

- Multiprocessor

- Message passing between nodes

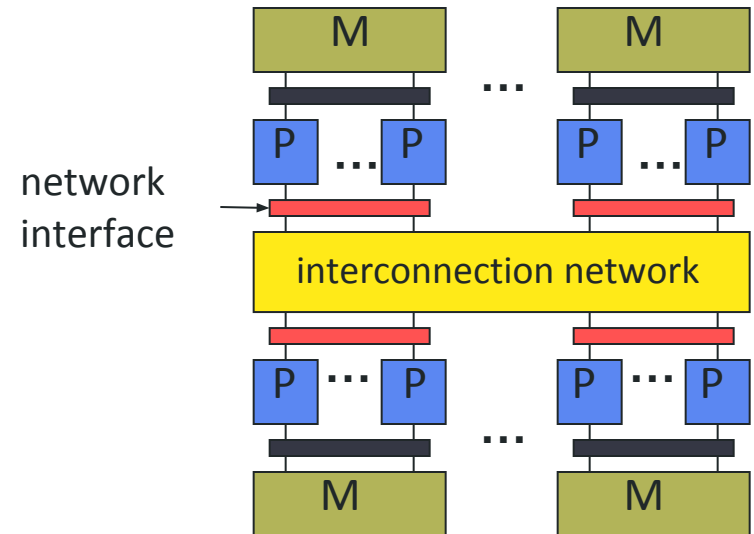


- Massively Parallel Processor (MPP)

- Many, many processors

- Cluster of SMPs

- Shared memory addressing within SMP node
 - Message passing between SMP nodes

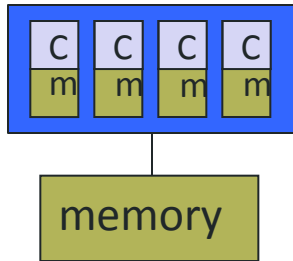


- Can also be regarded as MPP if processor number is large

Tipos de Arquitetura Paralelas - (3)

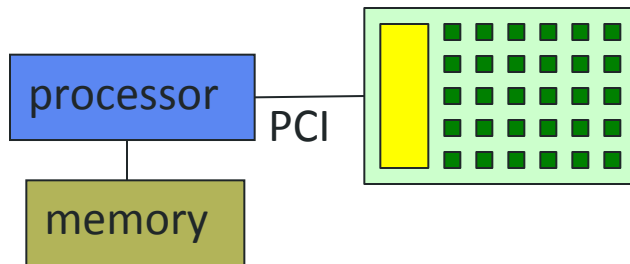
❑ Multicore

- Multicore processor

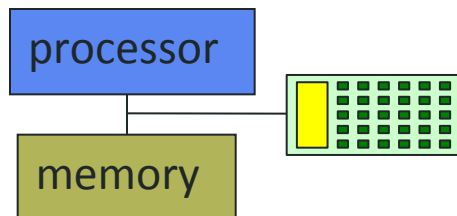


cores can be
hardware
multithreaded
(hyperthread)

- GPU accelerator

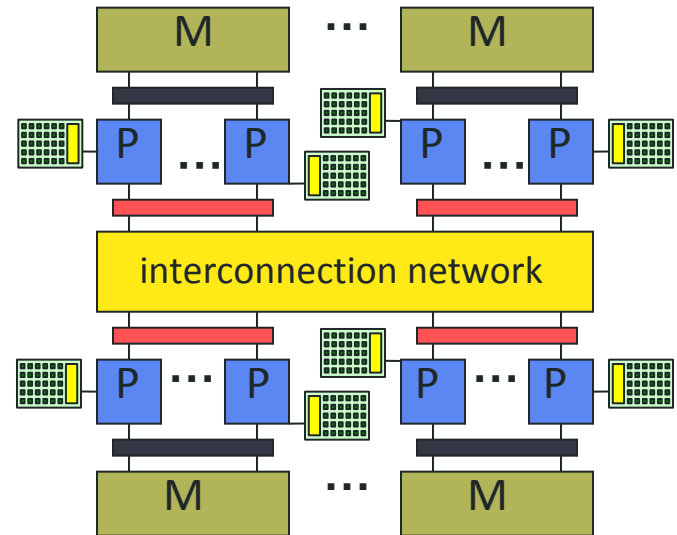


- “Fused” processor accelerator



- Multicore SMP+GPU Cluster

- Shared memory addressing within SMP node
- Message passing between SMP nodes
- GPU accelerators attached



Como conseguir paralelismo no hardware

- ❑ Paralelismo em nível de Instrução (ILP)
- ❑ Paralelismo de dados
 - Aumenta a quantidade de dados a ser operado ao mesmo tempo
- ❑ Processamento de processador
 - Aumenta o número de processadores
- ❑ Paralelismo no sistema de memória
 - Aumentar número de unidades de memória
 - Aumentar largura de banda para a memória
- ❑ Paralelismo na comunicação
 - Aumentar quantidade de interconexões entre os elementos
 - Aumentar largura de banda na comunicação

Paralelismo em nível de instrução

- ❑ Oportunidades para dividir o processamento da instrução
- ❑ Pipeline dentro da instrução
- ❑ Pipeline entre instruções
- ❑ Execução sobreposta
- ❑ Múltiplas unidades funcionais
- ❑ Execução fora de ordem
- ❑ Execução *multi-issue* (com as múltiplas unidades funcionais)
- ❑ Processador superscalar
- ❑ Superpipelining (muitos estágios, normalmente entre 2 a 26)
- ❑ Very Long Instruction Word (VLIW)
- ❑ Multithread em Hardware (hyperthreading)

Processamento Vetorial

❑ Processamento Escalar

- Instruções do processador operam sobre valores escalares
- Registradores inteiros e registradores de ponto flutuante

❑ Vetores

- Conjunto de dados escalares
- Registradores vetoriais
 - ◆ inteiro, ponto flutuante (tipicamente)
- Instruções vetoriais operam sobre registradores vetoriais (SIMD)

❑ Unidade de pipeline vetorial

❑ Múltiplas unidades vetoriais

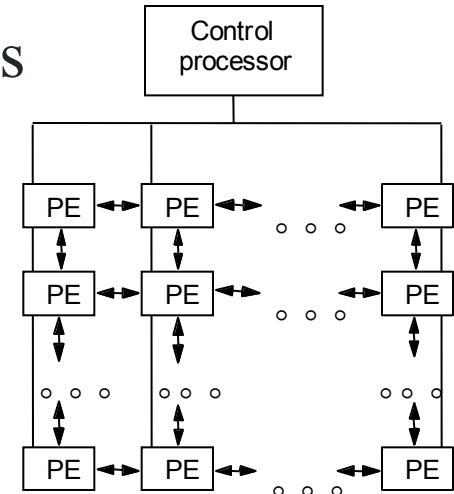
Liquid-cooled with inert fluorocarbon. (That's a waterfall fountain!!!)



Cray 2

Arquiteturas de Dados Paralelos

- ❑ SIMD (Single Instruction Multiple Data)
 - Thread única de controle lógico (instrução)
 - Processador associado com elementos de dados
- ❑ Arquitetura
 - Array processadores simples com memória
 - Processadores colocados em uma topologia regular
 - Processador controlador emite as instruções
 - ◆ Todos os processadores executam a mesma instrução
 - Sincronização e comunicação especializada
 - Operações de redução especializadas
 - Processamento em array

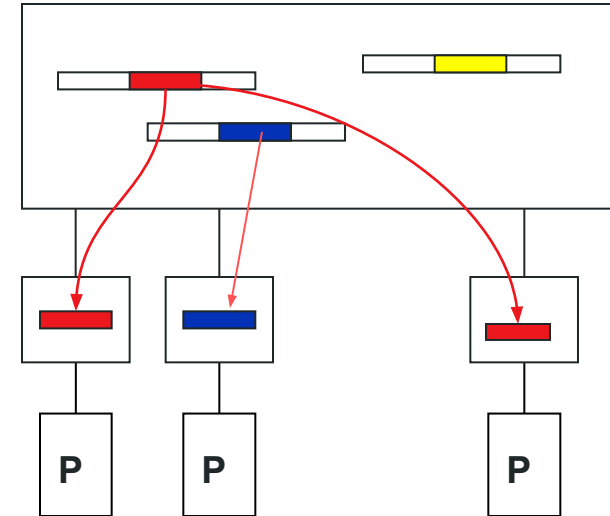


Memória Física Compartilhada

- ❑ Adiciona processadores a um sistema de processador único
- ❑ Processadores *compartilham* recursos do sistema computacional
 - Memória, armazenamento, ...
- ❑ Compartilhando a memória física
 - Qualquer processador pode referenciar qualquer localização da memória
 - Qualquer controlador de I/O pode referenciar qualquer localização da memória
 - Um único espaço de endereçamento da memória física
- ❑ Sistema Operacional roda em qualquer processador, ou em todos
 - SO enxerga um único espaço de endereçamento da memória física
 - Utiliza a memória compartilhada para coordenar
- ❑ Comunicação ocorre como resultado de cargas e armazenamentos, i.e., escritas e leituras na memória compartilhada

Cache em Sistemas de Memória Compartilhada

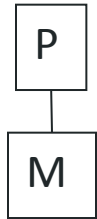
- ❑ Reduz latência média
 - replicação automática mais perto do processador
- ❑ Reduz largura de banda média
- ❑ Dados são transferidos logicamente da memória do produtor para a memória do consumidor
 - store reg → mem
 - load reg ← mem
- ❑ Processadores podem compartilhar dados eficientemente
- ❑ O que acontece com **load** e **store** são executados em diferentes processadores?
- ❑ Problemas de Coerência de Cache



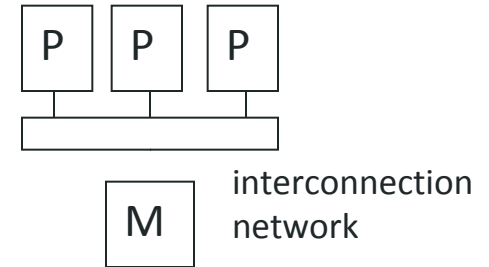
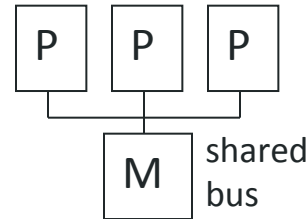
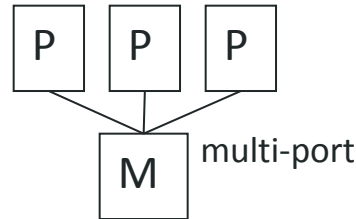
Shared Memory Multiprocessors (SMP)

❑ Tipos de Arquiteturas

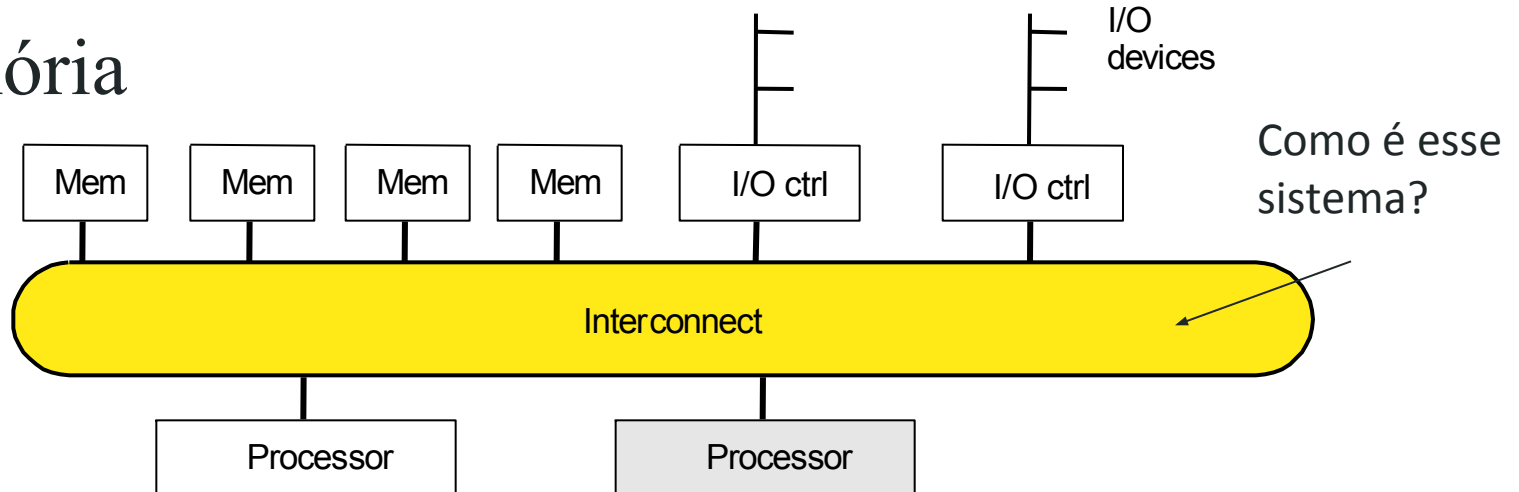
Processador único



Múltiplos processadores

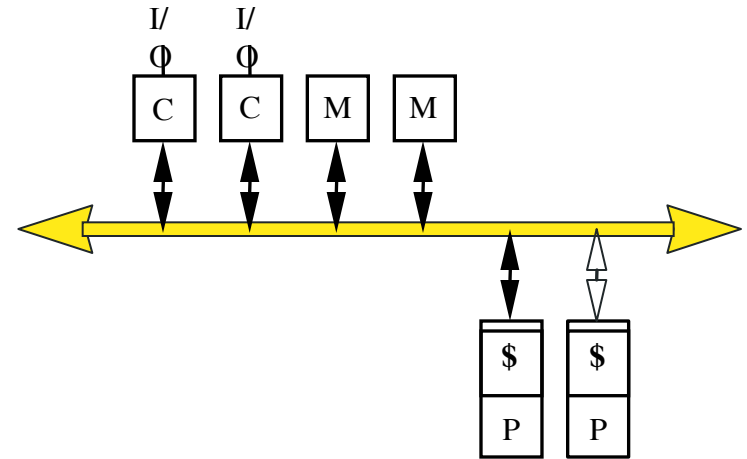


❑ Diferenças estão na interconexão do sistema de memória



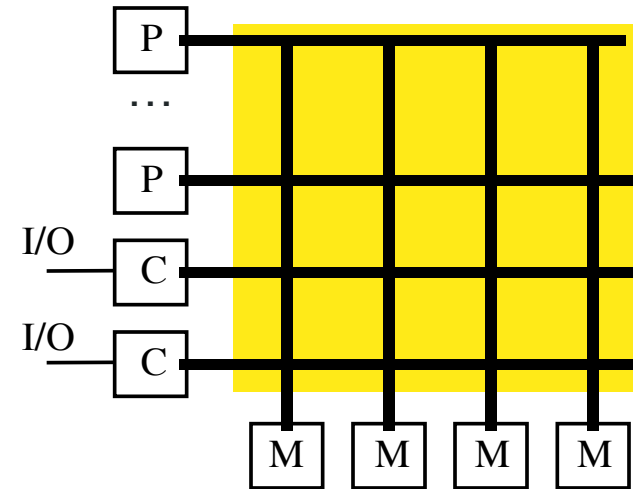
Bus-based SMP

- ❑ Bus de memória lida com todo o tráfego de leitura/escrita da memória
- ❑ Processadores compartilham bus
- ❑ *Uniform Memory Access (UMA)*
 - Memória (não o cache) uniformemente equidistante
 - Leva o mesmo tempo (geralmente) para acessar
- ❑ Pode ter múltiplos módulos de memória
 - Intercalação do espaço de endereçamento físico
- ❑ Cache introduz a hierarquia de memória
 - Leva a problemas de consistência de dados
 - Necessário hardware para coerência de cache (*CC-UMA*)



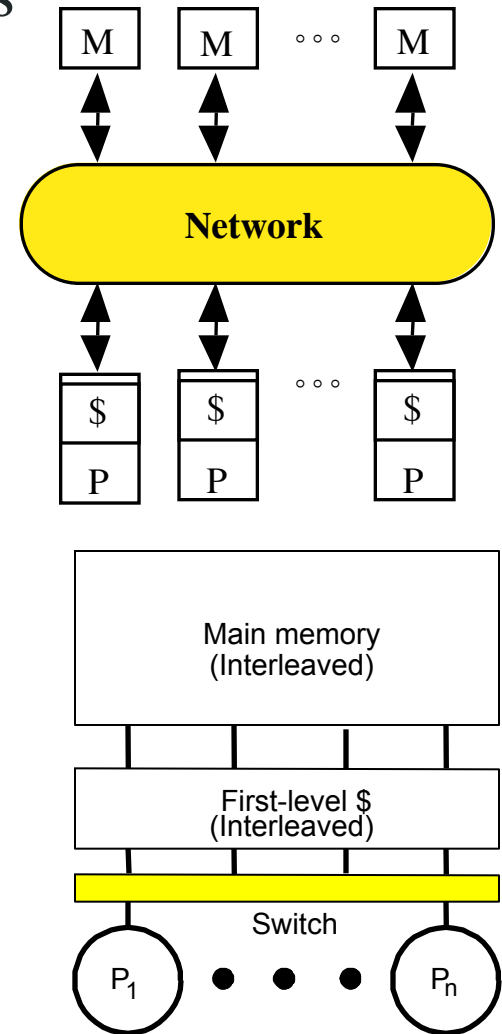
Crossbar SMP

- ❑ Replica o bus de memória para cada processador e controlador de I/O
 - Cada processador tem um caminho direto
- ❑ Arquitetura UMA SMP
- ❑ Ainda pode ter problemas de coerência de cache
- ❑ Memória intercalada ou multibanco
- ❑ Vantagens
 - Largura de banda escala linearmente (sem links compartilhados)
- ❑ Problemas
 - Alto custo incremental (não é viável para muitos processadores)
 - Utiliza rede de interconexão comutada multi-estágio

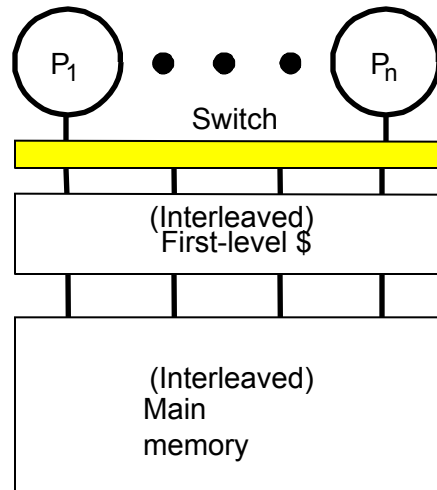


Cache Compartilhado e SMP “Dance Hall”

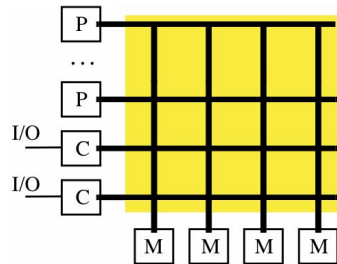
- ❑ Rede de interconexão conecta processadores à memória
- ❑ Memória Centralizada (UMA)
- ❑ Rede determina o desempenho
 - Continuum de bus para crossbar
 - Largura de banda da memória escalável
- ❑ Memória é fisicamente separada dos processadores
- ❑ Pode ter problemas de coerência de cache
- ❑ Cache compartilhado reduz problemas de coerência e provê granularidade fina no compartilhamento de dados



Natural Extensions of the Memory System

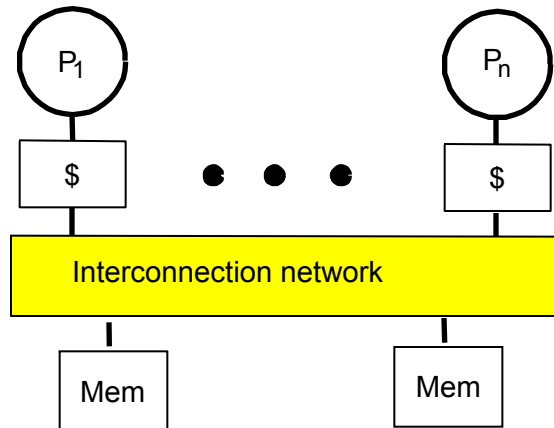


Shared Cache

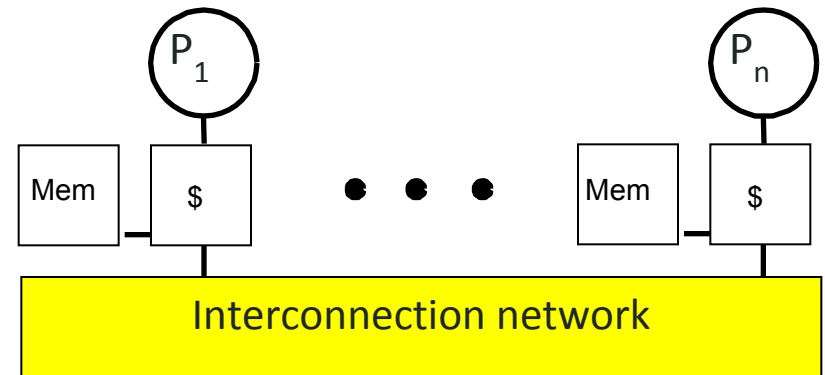


Crossbar, Interleaved

Scale →



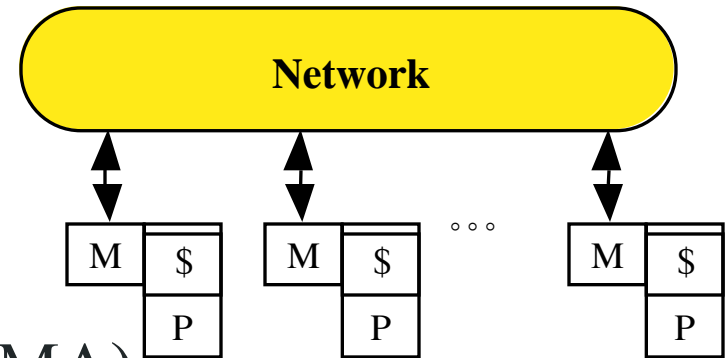
Centralized Memory
Dance Hall, UMA



Distributed Memory (NUMA)

SMPs Non-Uniform Memory Access (NUMA)

- ❑ Memória Distribuída
- ❑ Memória é fisicamente residente próxima a cada processador
- ❑ Memória ainda é compartilhada
- ❑ *Non-Uniform Memory Access (NUMA)*
 - Memória local e memória remota
 - Acesso à memória local é mais rápido, à memória remota é mais lento
 - Acesso não é uniforme
 - Desempenho dependerá da localidade dos dados
- ❑ Coerência de Cache ainda é um problema (mais sério)
- ❑ Arquitetura da rede de interconexão é mais escalável



Coerência de Cache e SMPs

- ❑ Caches tem papel fundamental no desempenho do SMP
 - Reduz tempo médio de acesso aos dados
 - Reduz a demanda por largura de banda colocada sobre o sistema de interconexão compartilhado
- ❑ Caches privados de cada processador criam um problema
 - Cópias de uma variável pode estar presente em múltiplos caches
 - Uma escrita por um processador pode não se tornar visível aos outros
 - ◆ Eles continuarão acessando o valor obsoleto em seus caches

⇒ *Problema de coerência de cache*
- ❑ O que fazemos a respeito?
 - Organizar a hierarquia de memória para fazê-la desaparecer
 - Detectar e tomar ações para eliminar o problema

Definições

- ❑ Operações de Memória (load, store, read-modify-write, ...)
 - ❑ A questão da memória é a operação apresentada do sistema de memória
 - ❑ Perspectiva do Processador
 - Write: leituras subsequentes retornam o valor
 - Read: escritas subsequentes não afetam o valor (já lido)
 - ❑ *Sistema de Memória Coerente*
 - Existe uma ordem serial das operações de memória em cada localização, de tal forma que:
 - ◆ operações emitidas por um processo aparecem na ordem emitida
 - ◆ o valor retornado por cada leitura é aquele escrito pela escrita anterior
- ⇒ propagação de escrita + serialização de escrita

Motivação para Consistência de Memória

- ❑ Coerência implica que escritas a uma localização se tornam visíveis a todos os processadores na mesma ordem
- ❑ Mas quando uma escrita se torna visível?
- ❑ Como estabelecemos ordem entre uma escrita e uma leitura por diferentes processadores?
 - Utilizar a sincronização de eventos
- ❑ Implementar protocolo de hardware para coerência de cache
- ❑ Protocolo será baseado no modelo de consistência de memória

P_1	P_2
<hr/>	
/* Assume initial value of A and flag is 0 */	
A = 1;	while (flag == 0) ; /* spin idly */
flag = 1;	print A;

Consistência de Memória

- ❑ Especifica as restrições na ordem na as operações de memória (de qualquer processo) podem aparecer para executar com relação uma à outra
 - Que ordens são preservadas?
 - Dada uma “leitura” (load), restringe os valores possíveis retornados por ela
- ❑ Implicações tanto para programadores como para projetistas do sistema
 - Programador a utiliza para pensar sobre corretude
 - Projetista de Sistema pode utilizar para restringir quantos acessos podem ser reordenados pelo compilador ou pelo hardware
- ❑ É um contrato entre o programador e o sistema

Consistência Sequencial

- ❑ Ordem total alcançada através da intercalação de acessos dos diferentes processos
 - Mantém a *ordem do programa*
 - Operações de memória (de todos os processos) aparecem para emitir, executar e completar atomicamente com relação às outras
 - Como se houvesse uma única memória (sem cache)

“A multiprocessor is sequentially consistent if the result of any execution is the same as if the operations of all the processors were executed in some sequential order, and the operations of each individual processor appear in this sequence in the order specified by its program.” [Lamport, 1979]

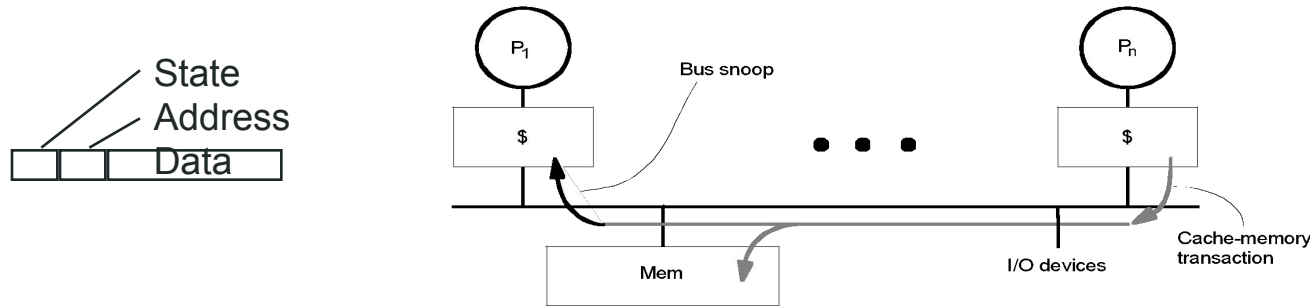
Consistência Sequencial (Condições Suficientes)

- ❑ Há uma ordem total consistente com as operações de memória que estão se tornando visíveis na ordem do programa
- ❑ Condições Suficientes
 - cada processo emite operações de memória na ordem do programa
 - após uma operação de escrita ser emitida, o processo emissor espera completar antes de emitir a próxima operação de memória (escritas atômicas)
 - após uma leitura ser emitida, o processo emissor espera pelo seu término e pela escrita cujo valor está sendo retornado também se completar (globalmente) antes de emitir sua próxima operação de memória.
- ❑ Arquiteturas com coerência de cache implementam consistência

Arquitetura de Cache Coerente (CC) baseada em bus

- ❑ Transações do Bus
 - Conjunto único de fios conecta vários dispositivos
 - Protocolo do Bus: arbitração, comando/endereço, dado
 - Cada dispositivo observa todas as transações
- ❑ Diagrama de transição de estados dos blocos do cache
 - Máquina de estados finitos especificam como a provisão dos blocos mudam
 - ◆ inválido, válido, sujo (dirty)
 - *Protocolo bisbilhoteiro (Snoopy)*
- ❑ Escolhas básicas
 - Write-through vs Write-back
 - Invalidar vs. Atualizar

Protocolo Snoopy de Coerência de Cache



- ❑ Bus é um meio de broadcast
- ❑ Cache sabe o que ele tem
- ❑ Controlador de cache “bisbilhota” todas as transações no bus compartilhado
 - transação é relevante se afeta algum bloco que está naquele cache
 - toma uma ação para assegurar coerência
 - ◆ invalidar, atualizar, ou suprir o valor
 - depende do estado do bloco e do protocolo

Exemplo: Invalida com Write-back

