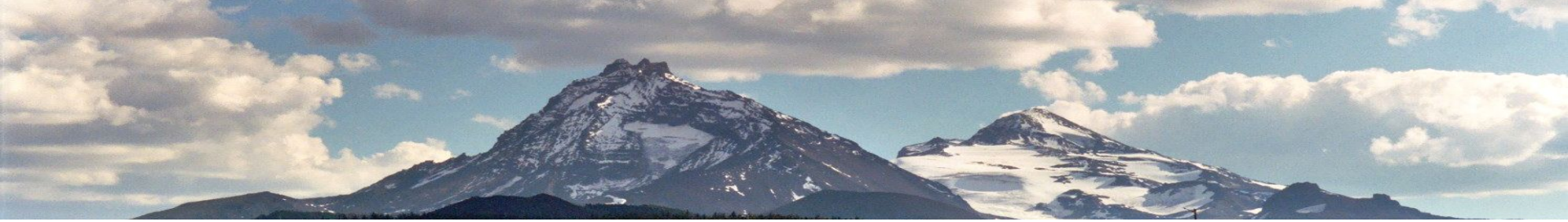


Arquitetura de Computadores Paralelos

Computação Paralela

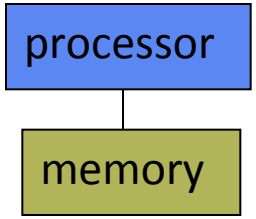
Faculdade de Computação e Informática

Outline

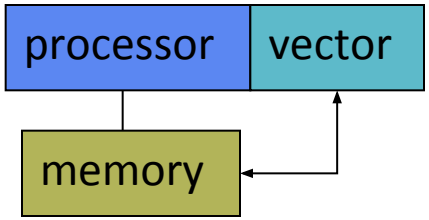
- Tipos de arquiteturas paralelas
- Paralelismo em nível de instrução
- Processamento vetorial
- SIMD
- Memória compartilhada
  - Organização de Memória: UMA, NUMA
- Interconnection networks
- Clusters

Tipos de Arquitetura Paralelas

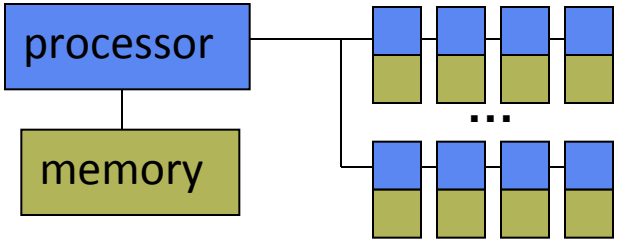
- Uniprocessor
  - Scalar processor



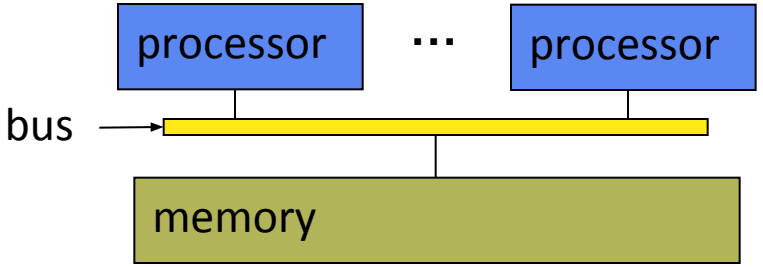
  - Vector processor



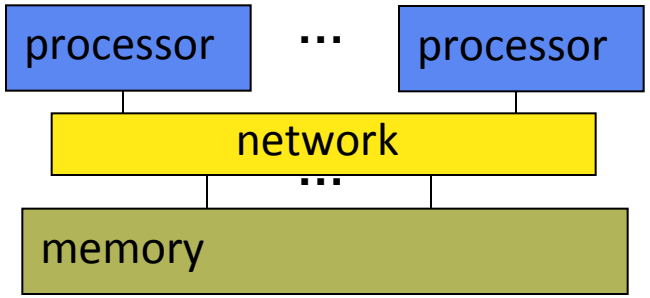
  - Single Instruction Multiple Data (SIMD)



- Shared Memory Multiprocessor (SMP)
  - Shared memory address space
  - Bus-based memory system

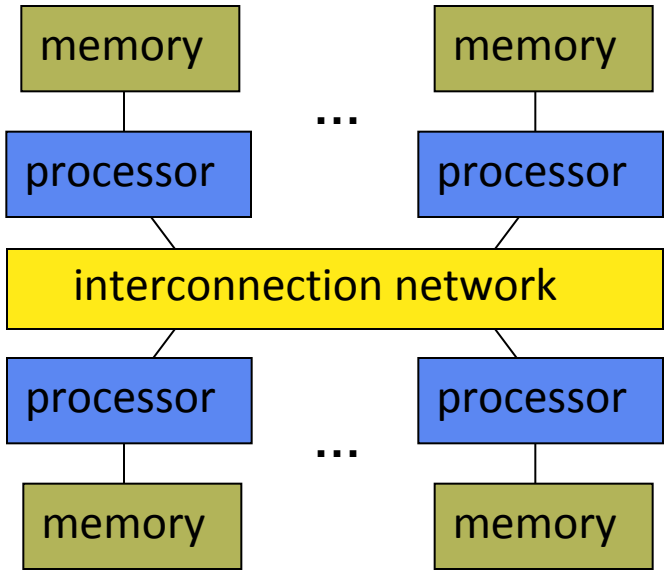


  - Interconnection network



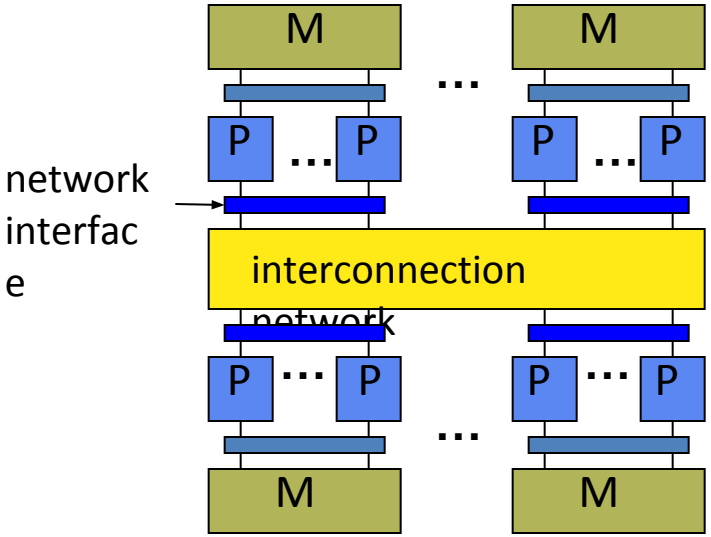
Tipos de Arquitetura Paralelas - (2)

- Distributed Memory Multiprocessor
  - Message passing between nodes



  - Massively Parallel Processor (MPP)
    - Many, many processors

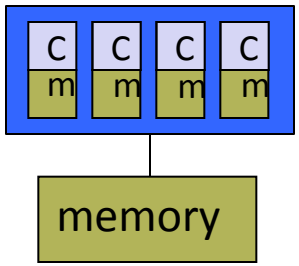
- Cluster of SMPs
  - Shared memory addressing within SMP node
  - Message passing between SMP nodes



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

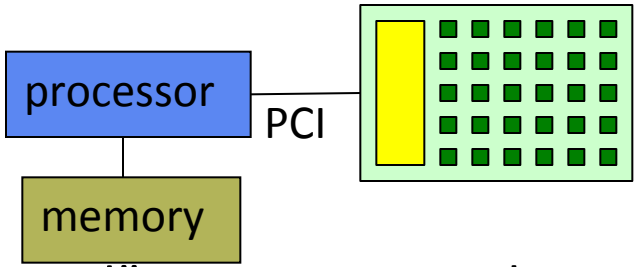
Tipos de Arquitetura Paralelas - (3)

- Multicore
  - Multicore processor

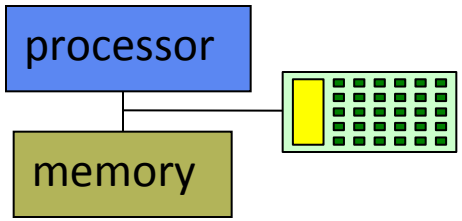


cores can be hardware multithreaded (hyperthread)

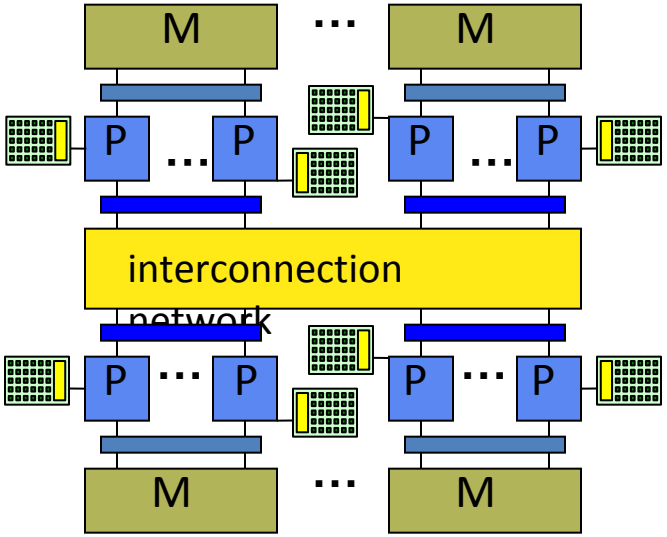
  - GPU accelerator



  - "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

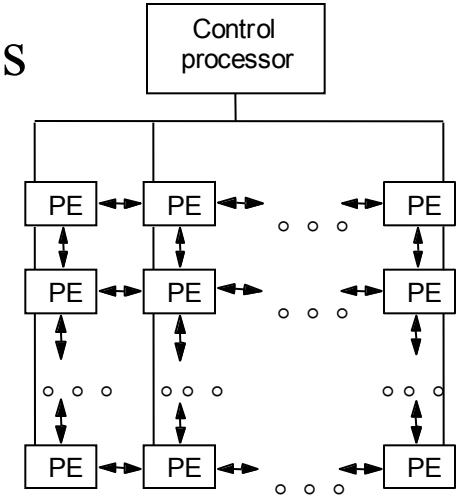
- Processamento Escalar
  - Instruções do processador operam sobre valores escalares
  - Registradores inteiros e registradores de ponto flutuante
- Vetores
  - Conjunto de dados escalares
  - Registradores vetoriais
    - inteiro, ponto flutuante (tipicamente)
  - Instruções vetoriais operam sobre registradores vetoriais (SIMD)
- Unidade de pipeline vetorial
- Múltiplas unidades vetoriais

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

Cray 2

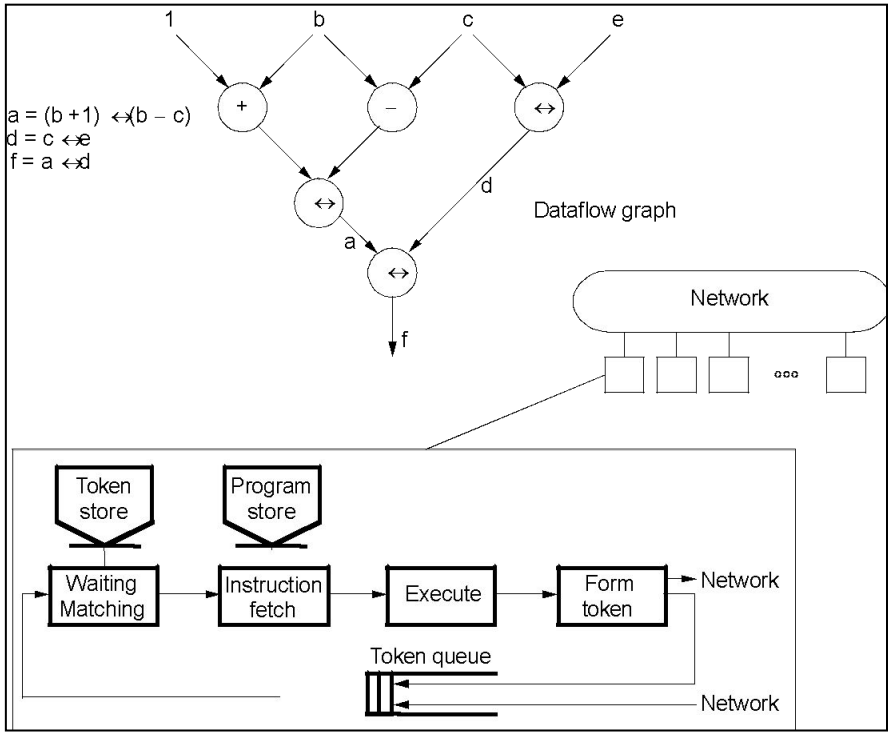
Arquiteturas de Dados Paralelos

- 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



Arquitetura de Dataflow

- Representa a computação como um grafo de dependências
- Operações são armazenadas em memória até que os operandos estejam prontos
- Operações podem ser despachadas aos processadores
- Tokens tem rótulos da próxima instrução p/ o processador
- Rótulo comparado no sistema de correspondência
- Uma correspondência dispara uma execução
- A máquina faz o trabalho duro de paralelização
- Difícil de construir corretamente

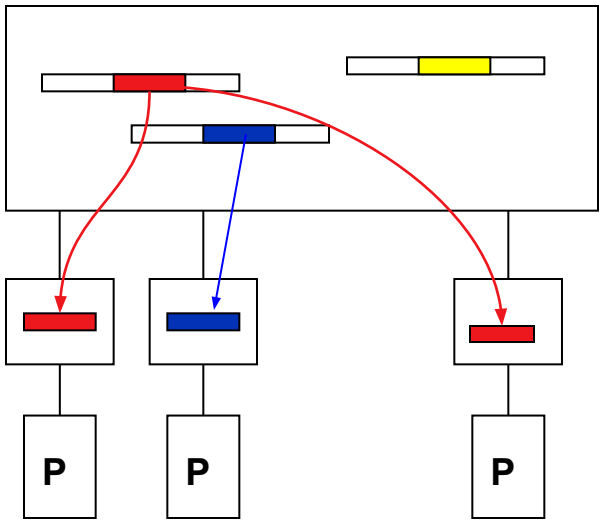


Memória Física Compartilhada

- 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

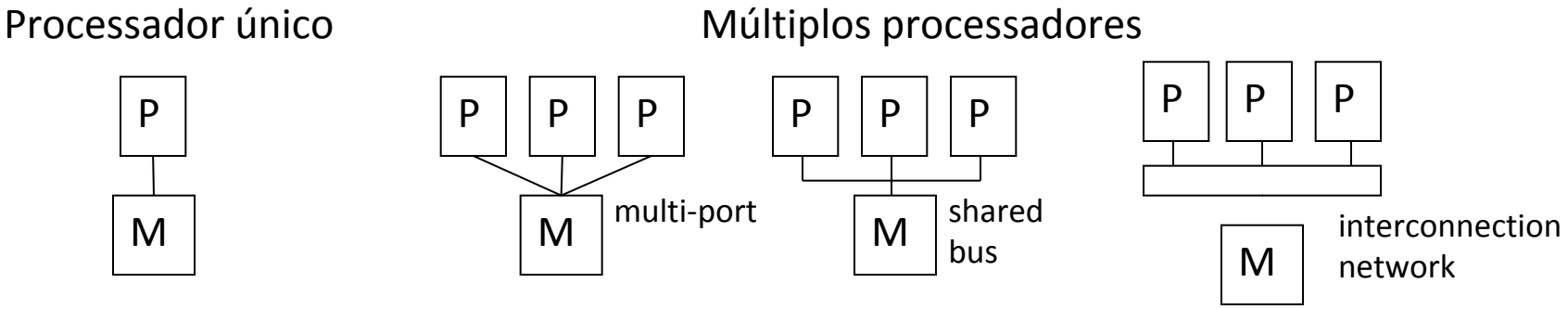
- Reduz latência média
  - replicação automática mais perto do processador
- Reduz largura de banda média
- Dados são transferidos logicamente da memória do produtor para a memória do consumidor
  - store reg → mem
  - load reg ← mem



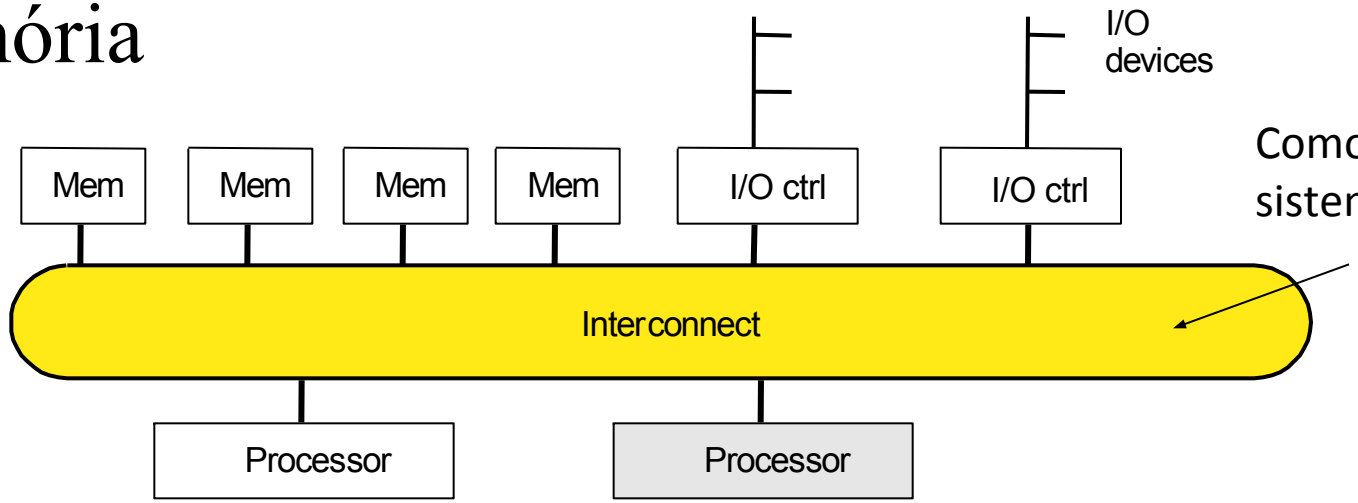
- Processadores podem compartilhar dados eficientemente
- O que acontece com **load** e **store** são executados em diferentes processadores?
- Problemas de Coerência de Cache

Shared Memory Multiprocessors (SMP)

- Tipos de Arquiteturas

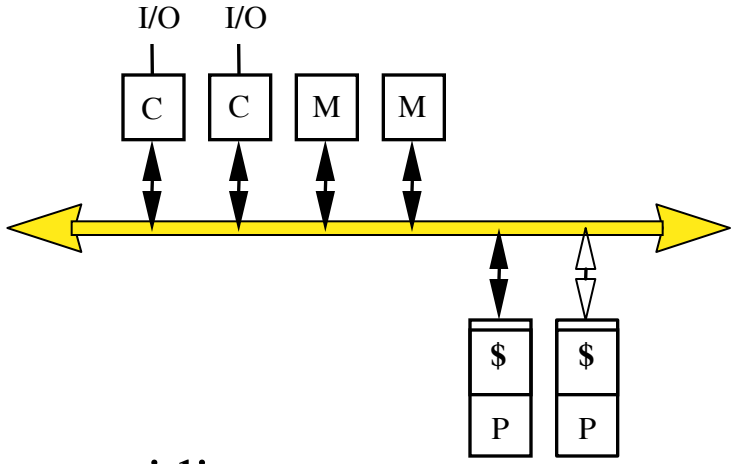


- Diferenças estão na interconexão do sistema de memória

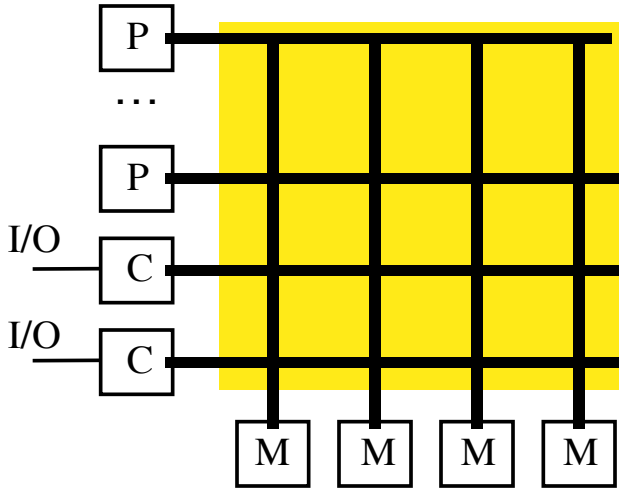


Bus-based SMP

- Bus de memória lida com todo o tráfico 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*)



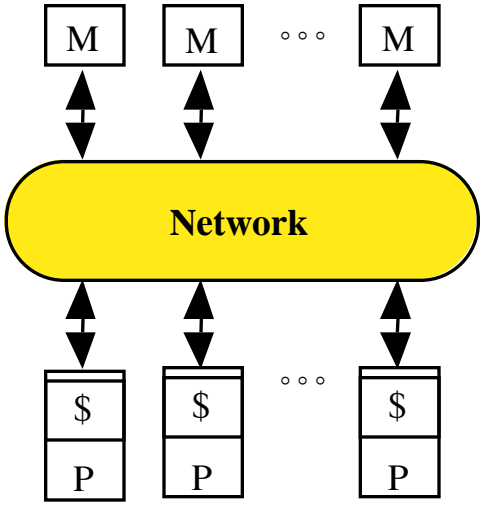
Crossbar SMP

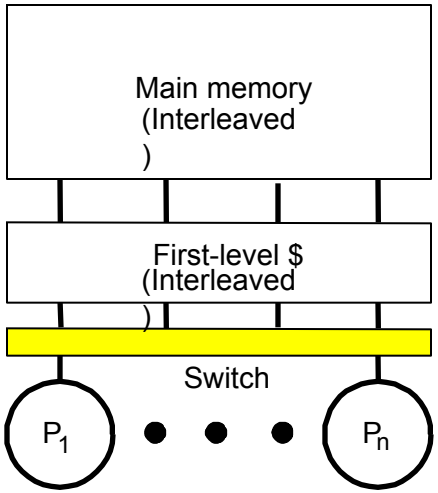


- Replica o bus de memória para cada processador e controlador de I/O
  - Cada processador tem um caminho direto
- Arquitetura UMA SMP
- Ainda pode ter problemas de coerência de cache
- Memória intercalada ou múlti-banco
- Vantagens
  - Largura de banda escala linearmente (sem links compartilhados)
-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

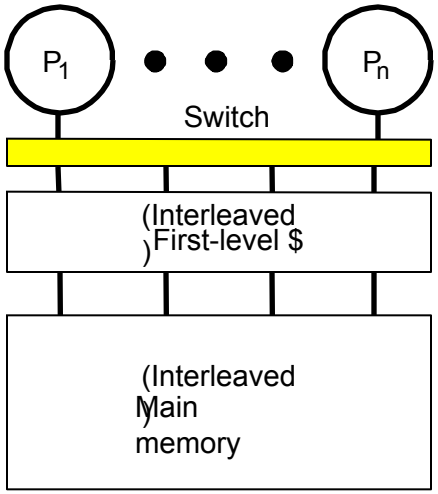
Cache Compartilhado e SMP "Dance Hall"

- 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

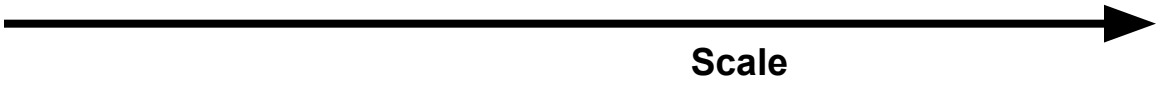


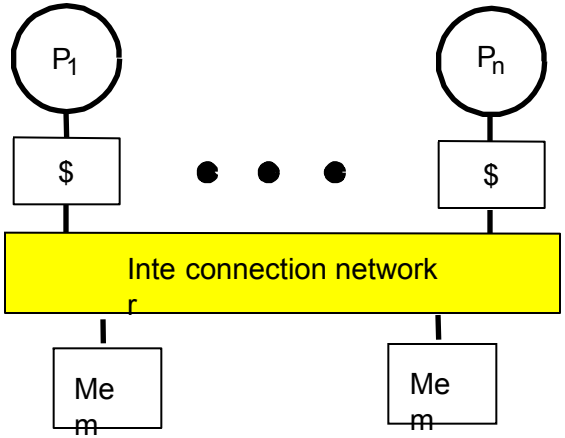


Natural Extensions of the Memory System

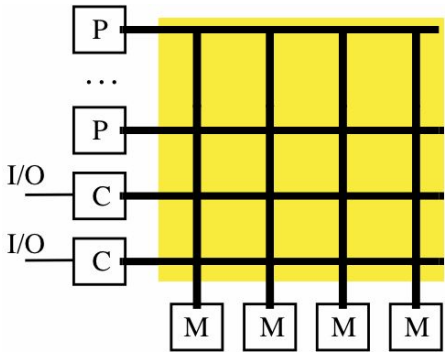


Shared Cache

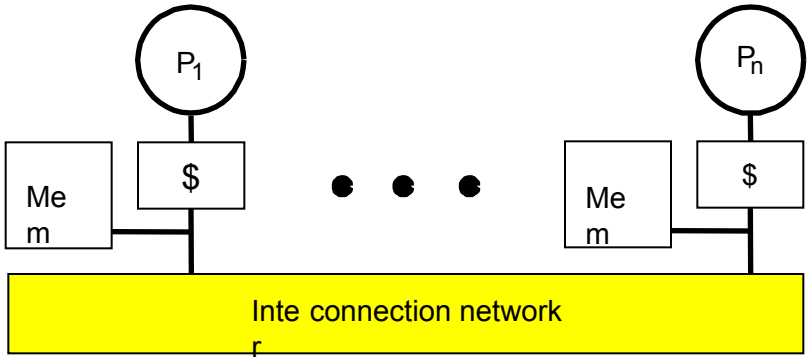




Centralized Memory
Dance Hall, UMA



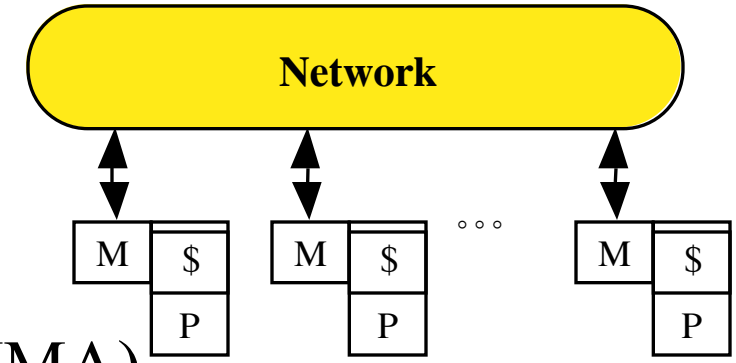
Crossbar, Interleaved



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 valur
  - 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

  $\Rightarrow$ 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
/* 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 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]

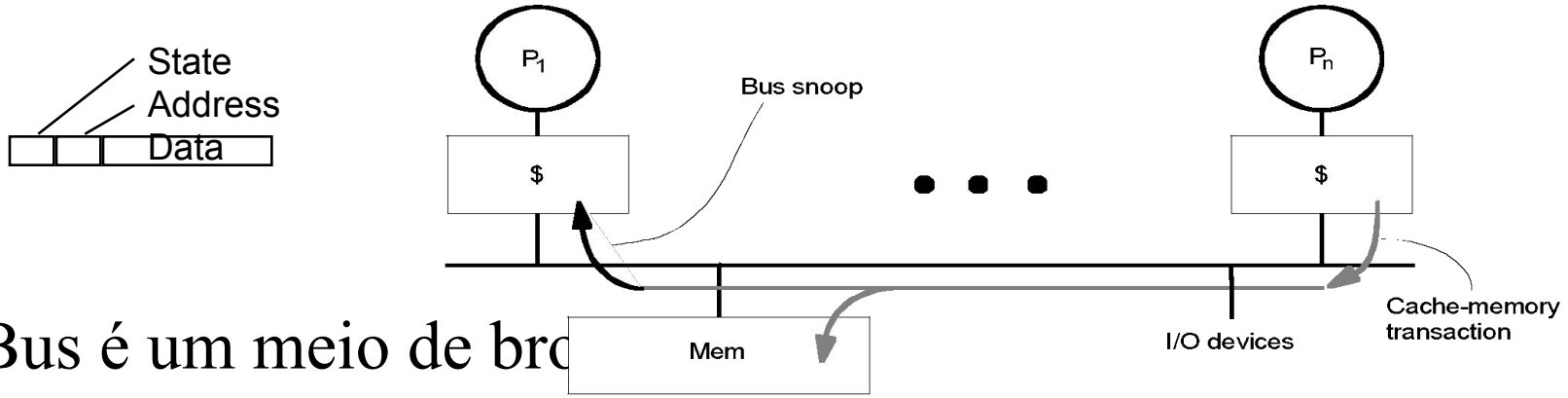
Consistência Sequencial (Condições Suficientes)

- 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

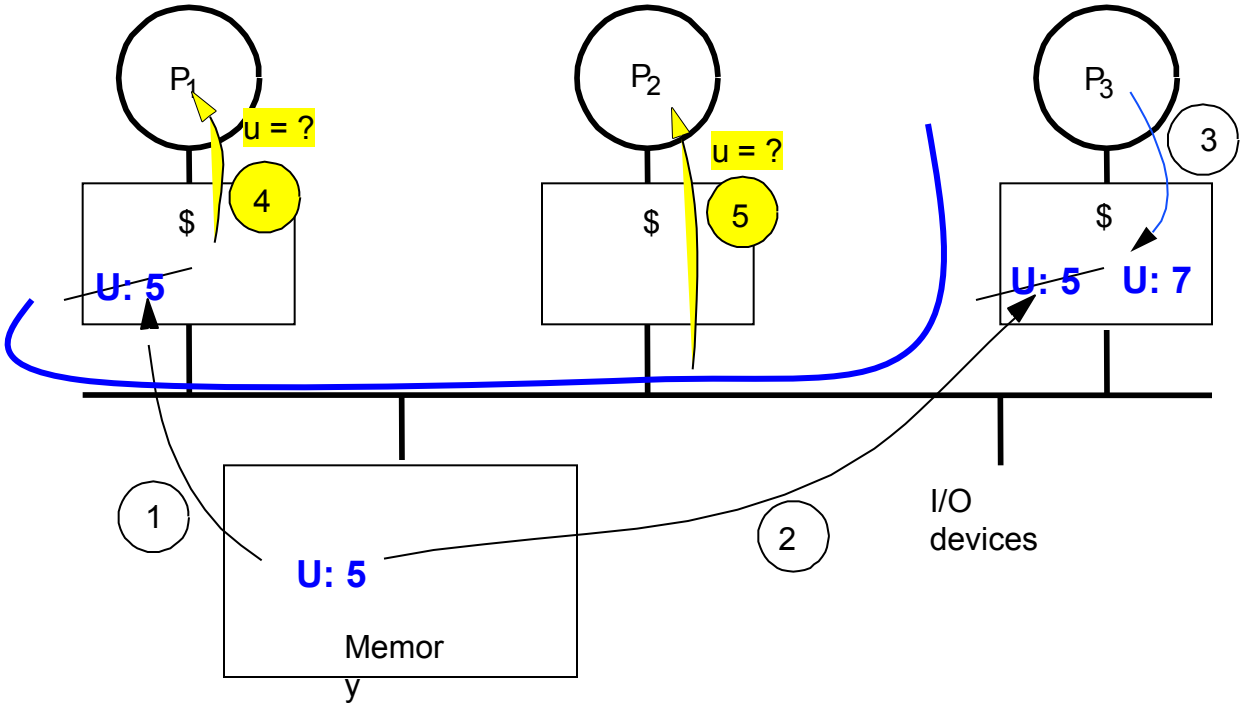
- Transações do Bus
  - Conjunto único de fios conecta vários dispositivos
  - Protocolo do Bus: arbitração, comando/endereço, dado
  - Cada dispositivo observa todas as transações
- Diagrama de transição de estados dos blocos do cache
  - Máquina de estados finitos especificam como a provisão dos blocos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

Protocolo Snoopy de Coerência de Cache

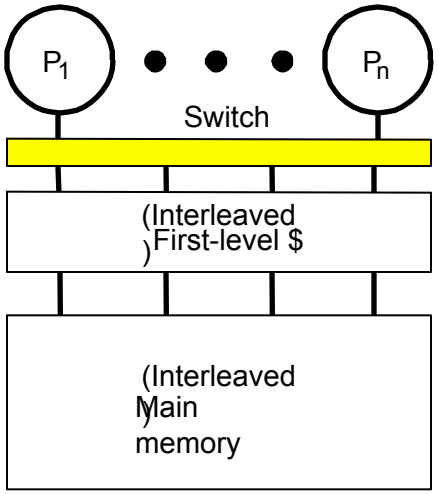


- Bus é um meio de bro
- Cache sabe o que ele tem
- Controlador de cache "bisbilhota" todas as transações no bus compartilhado
  - transação é relevante se afeta algum bloco que está naquele cache
  - toma uma ação para assegurar coerência
    - invalidar, atualizar, ou suprir o valor
  - depende do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

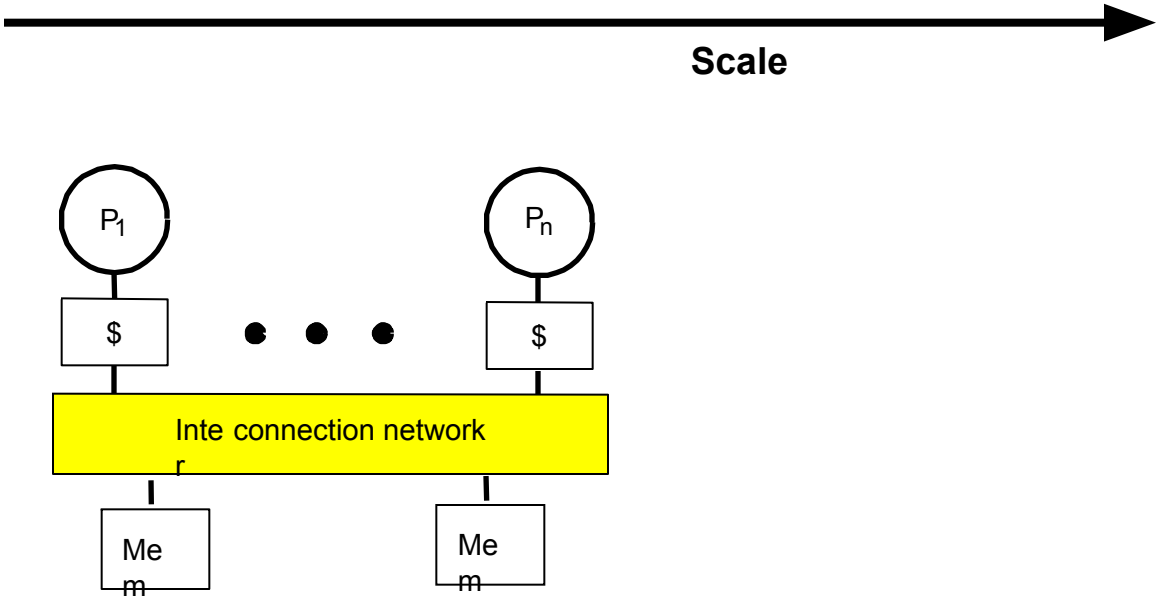
Exemplo: Invalida com Write-back



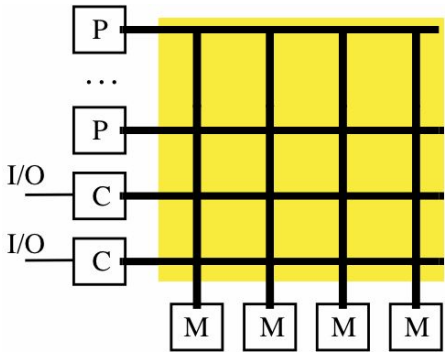
Natural Extensions of the Memory System



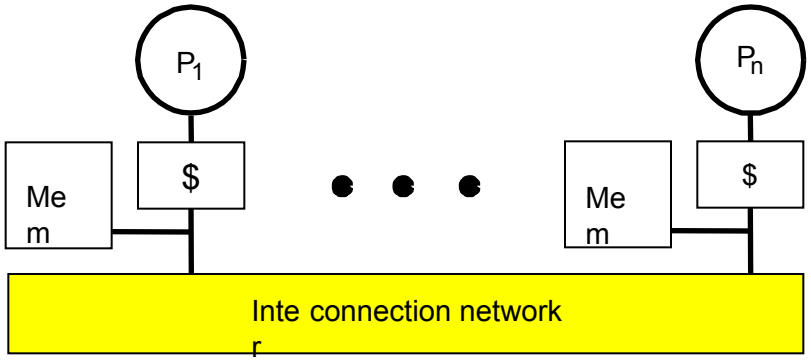
Shared Cache



Centralized Memory
Dance Hall, UMA



Crossbar, Interleaved



Distributed Shared Memory (NUMA)

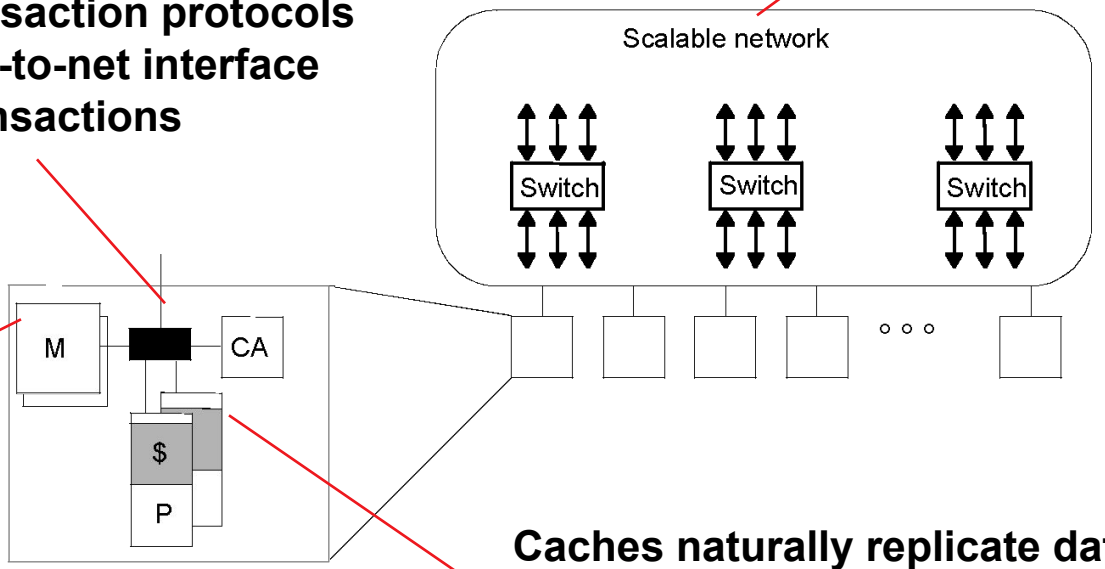
Memory Consistency

- Specifies constraints on the order in which memory operations (from any process) can appear to execute with respect to each other
  - What orders are preserved?
  - Given a load, constrains the possible values returned by it
- Implications for both programmer and system designer
  - Programmer uses to reason about correctness
  - System designer can use to constrain how much accesses can be reordered by compiler or hardware
- Contract between programmer and system
- Need coherency systems to enforce memory consistency

Context for Scalable Cache Coherence

Scalable Networks
- many simultaneous transactions

Realizing programming models through net transaction protocols
- efficient node-to-net interface
- interprets transactions

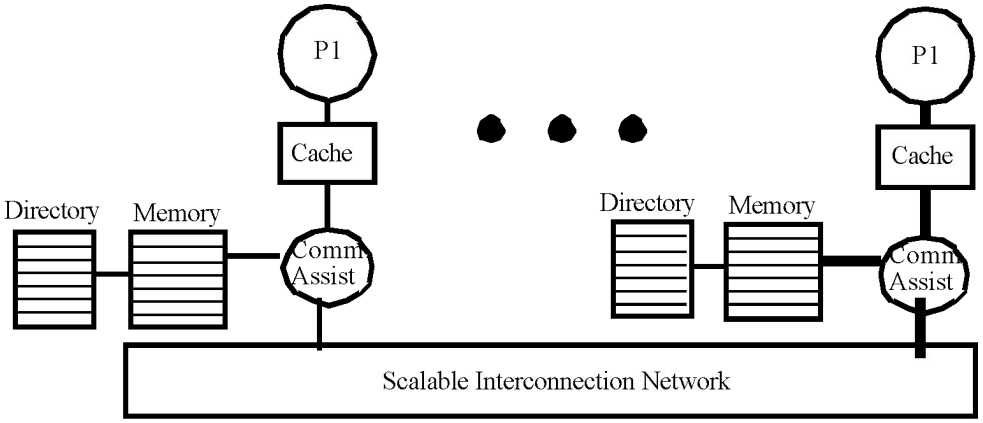


Scalable distributed memory

Caches naturally replicate data
- coherence through bus
- snooping protocols
- consistency

Need cache coherence protocols that scale!
- no broadcast or single point of order

Generic Solution: Directories



- Maintain state vector explicitly
  - associate with memory block
  - records state of block in each cache
- On miss, communicate with directory
  - determine location of cached copies
  - determine action to take
  - conduct protocol to maintain coherence

Requirements of a Cache Coherent System

- Provide set of states, state transition diagram, and actions
- Manage coherence protocol
  - (0) Determine when to invoke coherence protocol
  - (a) Find info about state of block in other caches to determine action
    - whether need to communicate with other cached copies
  - (b) Locate the other copies
  - (c) Communicate with those copies (inval/update)
- (0) is done the same way on all systems
  - state of the line is maintained in the cache
  - protocol is invoked if an "access fault" occurs on the line
- Different approaches distinguished by (a) to (c)

Bus-base Cache Coherence

- All of (a), (b), (c) done through broadcast on bus
  - faulting processor sends out a “search”
  - others respond to the search probe and take necessary action
- Could do it in scalable network too
- Conceptually simple, but broadcast doesn’t scale
  - on bus, bus bandwidth doesn’t scale
  - on scalable network, every fault leads to at least p network transactions
- Scalable coherence:
  - can have same cache states and state transition diagram
  - different mechanisms to manage protocol

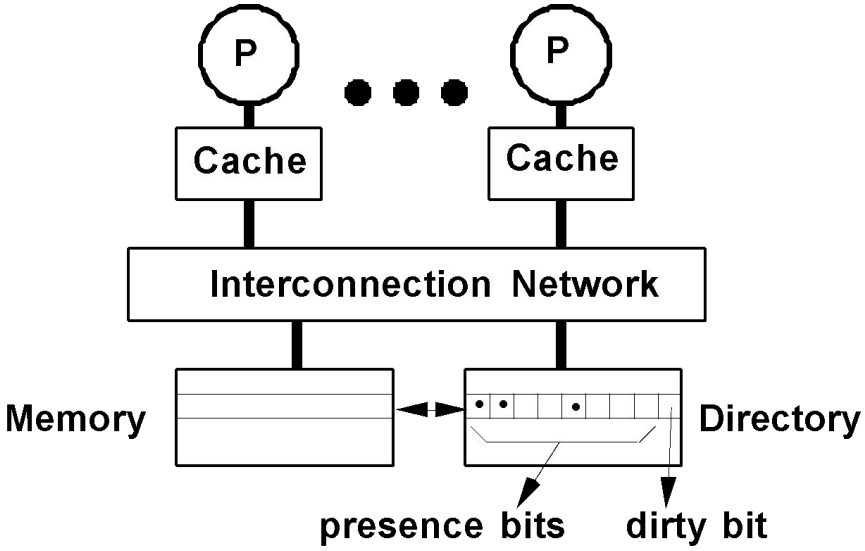
Basic Snoop Protocols

- Write Invalidate :
  - Multiple readers, single writer
  - Write to shared data: an invalidate is sent to all caches
  - Read Miss:
    - Write-through: memory is always up-to-date
    - Write-back: snoop in caches to find most recent copy
- Write Broadcast (typically write through):
  - Write to shared data: broadcast on bus, snoop and update
- Write serialization: bus serializes requests!
- Write Invalidate versus Broadcast

Scalable Approach: Directories

- Every memory block has associated directory information
  - keeps track of copies of cached blocks and their states
  - on a miss, find directory entry, look it up, and communicate only with the nodes that have copies if necessary
  - in scalable networks, communication with directory and copies is through network transactions
- Many alternatives for organizing directory information

Basic Operation of Directory

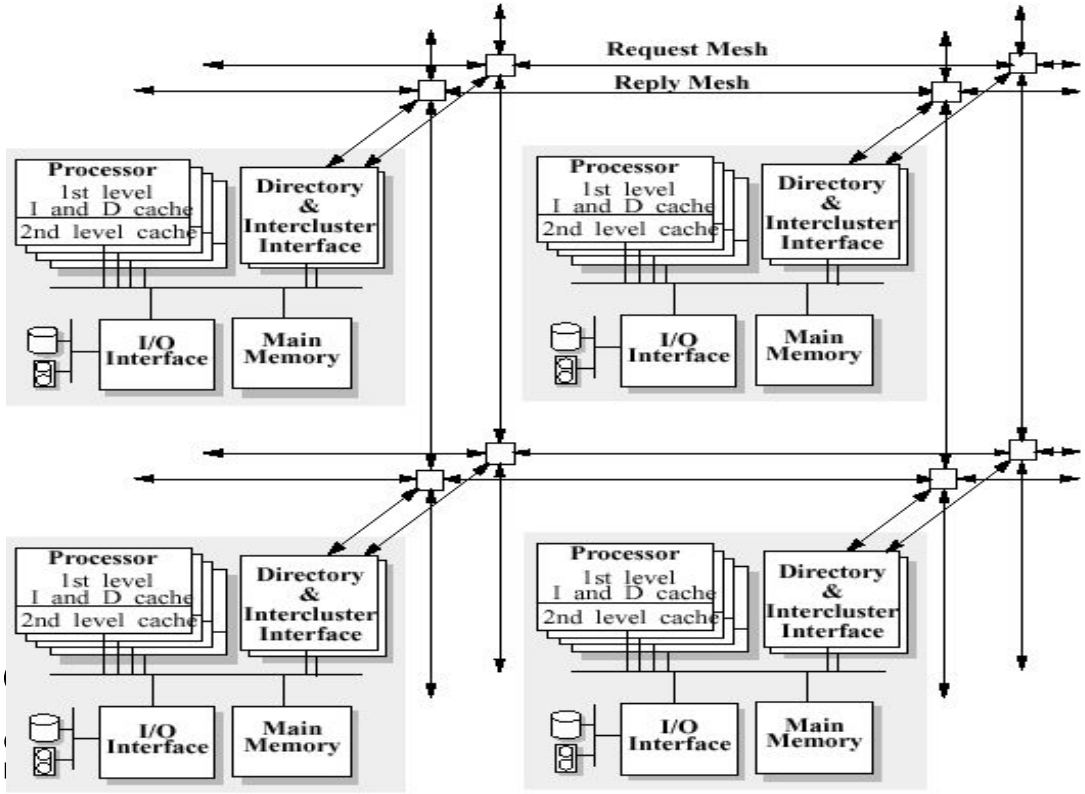


- k processors
- Each cache-block in memory
  - k presence bits and 1 dirty bit
- Each cache-block in cache
  - 1 valid bit and 1 dirty (owner) bit

- Read from memory
  - Dirty bit OFF
  - Dirty bit ON
- Write to memory
  - Dirty bit OFF

DASH Cache-Coherent SMP

- Directory Architecture for Shared Memory
- Stanford research project (early 1990s) for studying how to build cache-coherent shared memory architectures
- Directory-based cache c
- D. Lenoski, et al., "The
  Computer, Volume 25 Issue 3, pp: 63-79, March 1992

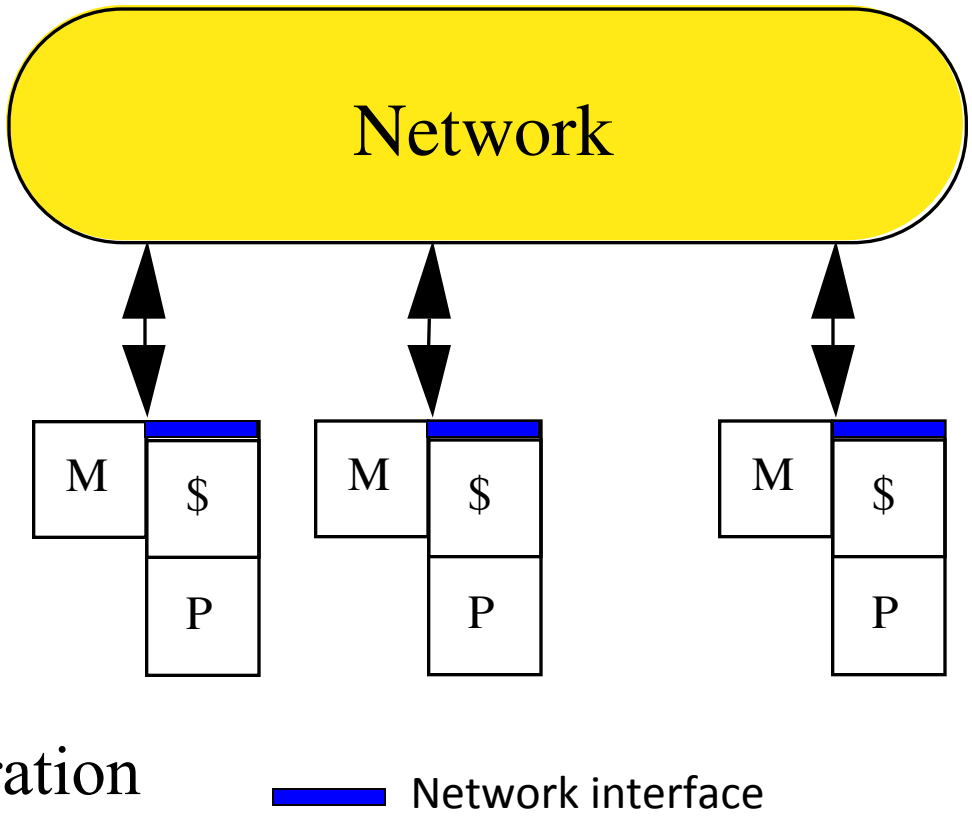


Distributed Memory Multiprocessors

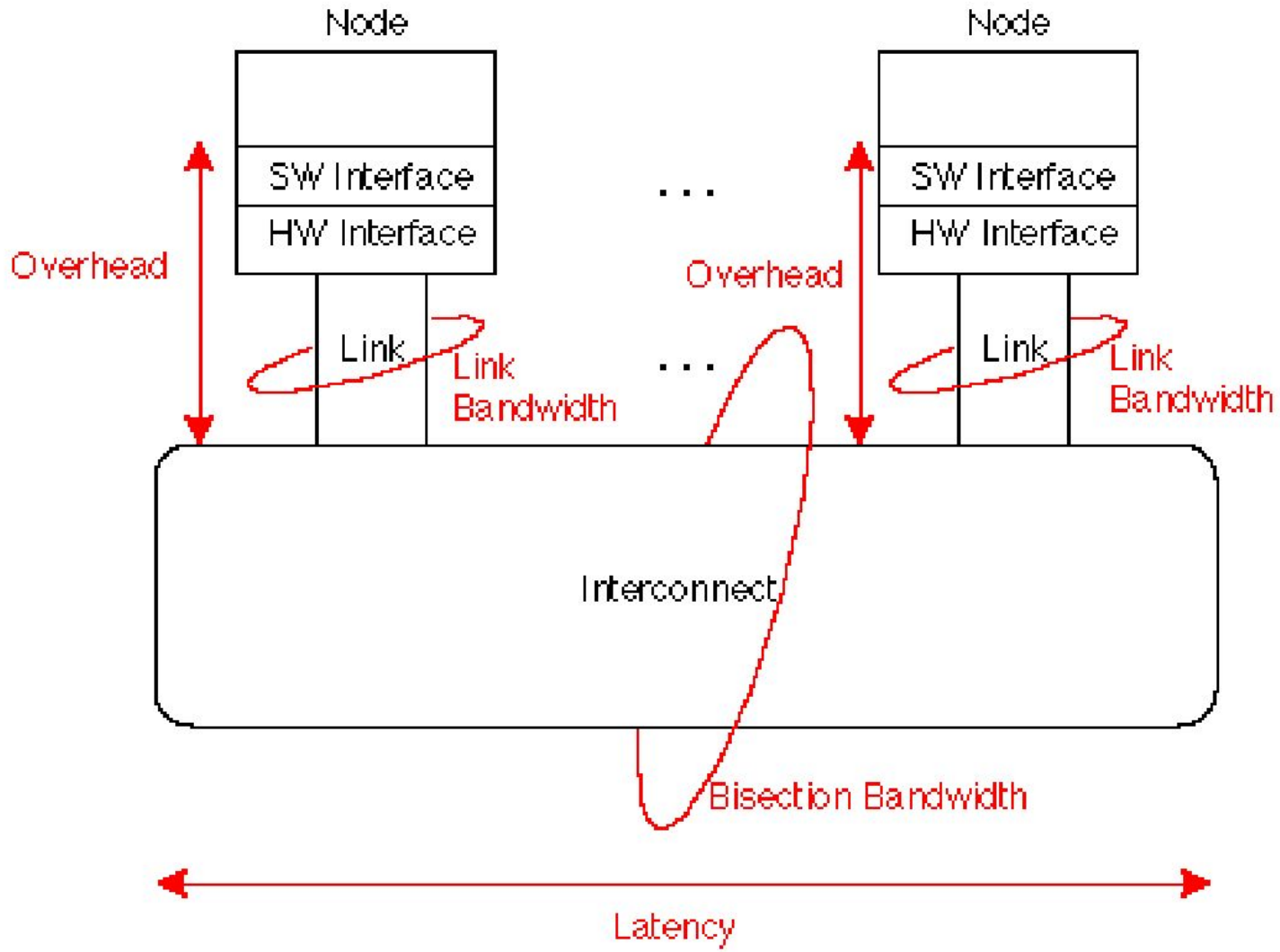
- Each processor has a local memory
  - Physically separated memory address space
- Processors must communicate to access non-local data
  - Message communication (message passing)
    - *Message passing architecture*
  - Processor interconnection network
- Parallel applications must be partitioned across
  - Processors: execution units
  - Memory: data partitioning
- Scalable architecture
  - Small incremental cost to add hardware (cost of node)

Distributed Memory (MP) Architecture

- Nodes are complete computer systems
  - Including I/O
- Nodes communicate via interconnection network
  - Standard networks
  - Specialized networks
- Network interfaces
  - Communication integration
- Easier to build



Network Performance Measures



Overhead: latency of interface vs. **Latency**: network

Performance Metrics: Latency and Bandwidth

- Bandwidth
  - Need high bandwidth in communication
  - Match limits in network, memory, and processor
  - Network interface speed vs. network bisection bandwidth
- Latency
  - Performance affected since processor may have to wait
  - Harder to overlap communication and computation
  - Overhead to communicate is a problem in many machines
- Latency hiding
  - Increases programming system burden
  - Examples: communication/computation overlaps, prefetch

Scalable, High-Performance Interconnect

- Interconnection network is core of parallel architecture
- Requirements and tradeoffs at many levels
  - Elegant mathematical structure
  - Deep relationship to algorithm structure
  - Hardware design sophistication
- Little consensus
  - Performance metrics?
  - Cost metrics?
  - Workload?
  - …

What Characterizes an Interconnection Network?

- Topology (what)
  - Interconnection structure of the network graph
- Routing Algorithm (which)
  - Restricts the set of paths that messages may follow
  - Many algorithms with different properties
- Switching Strategy (how)
  - How data in a message traverses a route
  - *circuit switching* vs. *packet switching*
- Flow Control Mechanism (when)
  - When a message or portions of it traverse a route
  - What happens when traffic is encountered?

Message Passing Model

- Hardware maintains send and receive message buffers
- Send message (synchronous)
  - Build message in local message send buffer
  - Specify receive location (processor id)
  - Initiate send and wait for receive acknowledge
- Receive message (synchronous)
  - Allocate local message receive buffer
  - Receive message byte stream into buffer
  - Verify message (e.g., checksum) and send acknowledge
- Memory to memory copy with acknowledgement and pairwise synchronization

Advantages of Shared Memory Architectures

- Compatibility with SMP hardware
- Ease of programming when communication patterns are complex or vary dynamically during execution
- Ability to develop applications using familiar SMP model, attention only on performance critical accesses
- Lower communication overhead, better use of BW for small items, due to implicit communication and memory mapping to implement protection in hardware, rather than through I/O system
- HW-controlled caching to reduce remote communication by caching of all data, both shared and private

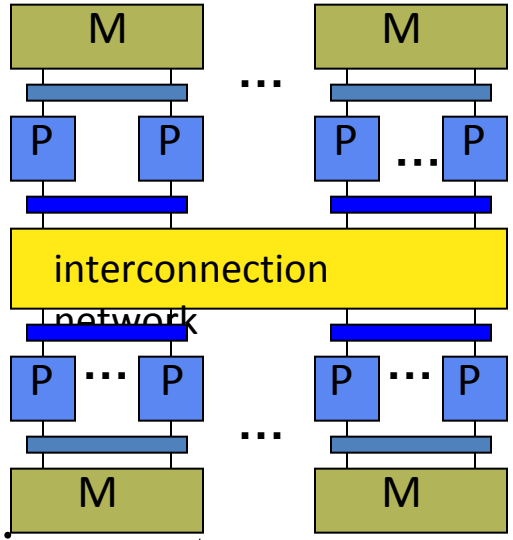
Computação Paralela - ICMC - USP

Advantages of Distributed Memory Architectures

- The hardware can be simpler (especially versus NUMA) and is more scalable
- Communication is explicit and simpler to understand
- Explicit communication focuses attention on costly aspect of parallel computation
- Synchronization is naturally associated with sending messages, reducing the possibility for errors introduced by incorrect synchronization
- Easier to use sender-initiated communication, which may have some advantages in performance

Clusters of SMPs

- Clustering
  - Integrated packaging of nodes
- Motivation
  - Ammortize node costs by sharing packaging and resources
  - Reduce network costs
  - Reduce communications bandwidth requirements
  - Reduce overall latency
  - More parallelism in a smaller space
  - Increase node performance
- Scalable parallel systems today are built as SMP clusters



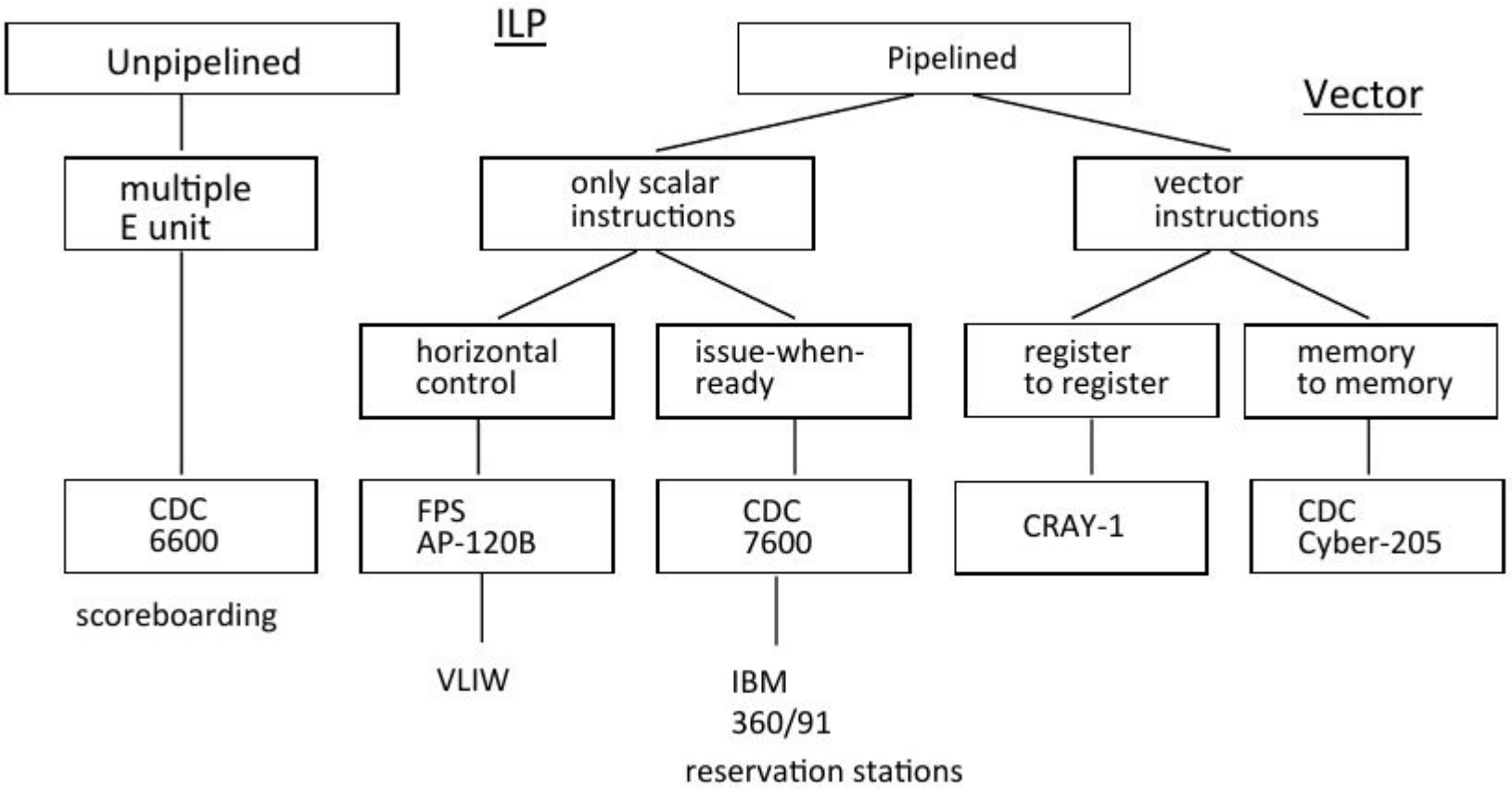
Next Class

- Parallel performance models

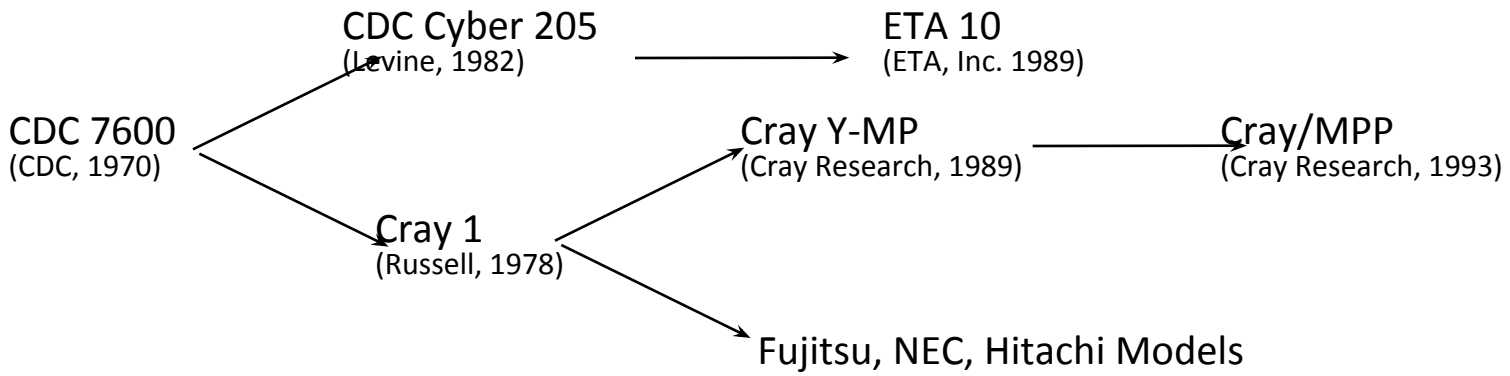
Slides retirados - para consulta

Paralelismo em Computadores de um processador

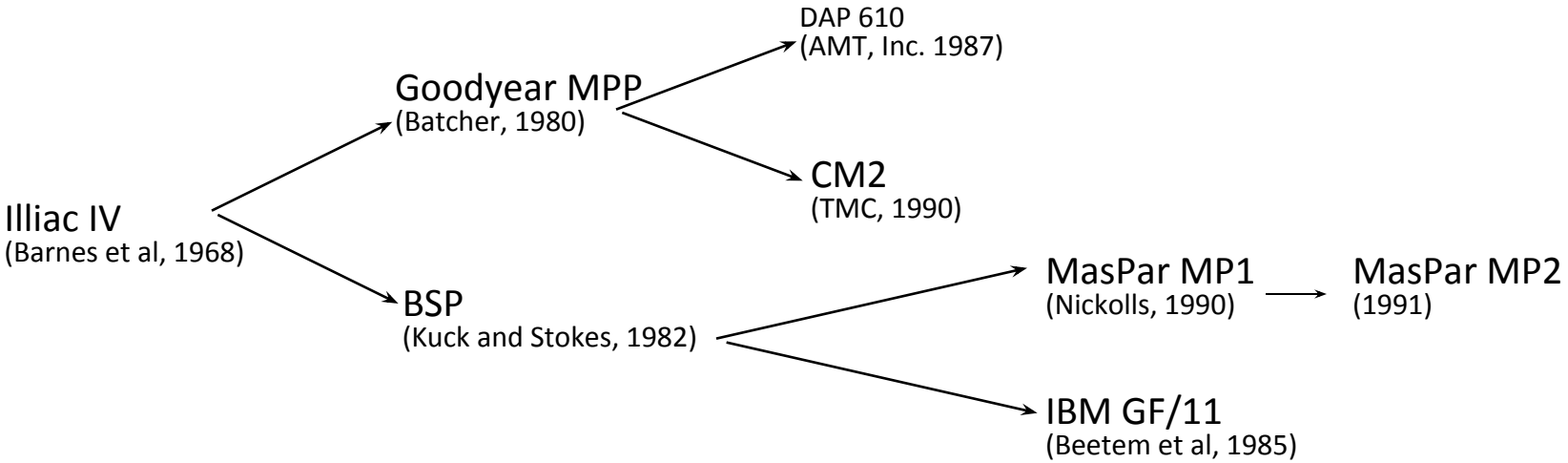
- História da inovação na arquitetura do processador



Timeline de Processamento Vetorial e SIMD



(a) Multivector track



(b) SIMD track