Arquiteturas Parelelas

Gonzalo Travieso

2020

Fases de paralelismo

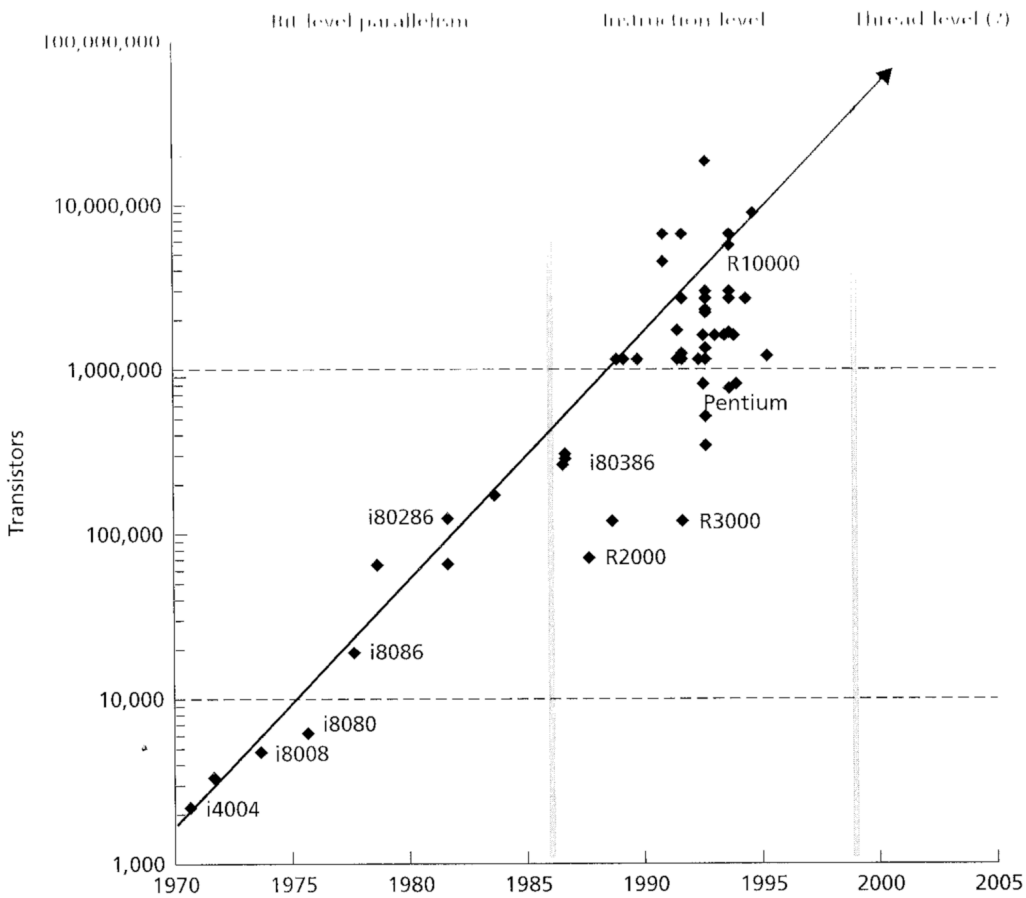


FIGURE 1.6 Number of transistors per processor chip over the last 25 years. The growth essentially follows Moore's Law, which says that the number of transistors doubles every two years. Forecasting from past trends, we can reasonably expect to be designing for a 50- to 100-million-transistor budget at the end of the decade. Also indicated are the epochs of design within the fourth, or VLSI, generation of computer architecture, reflecting the increasing level of parallelism.

Limitações de superescalares

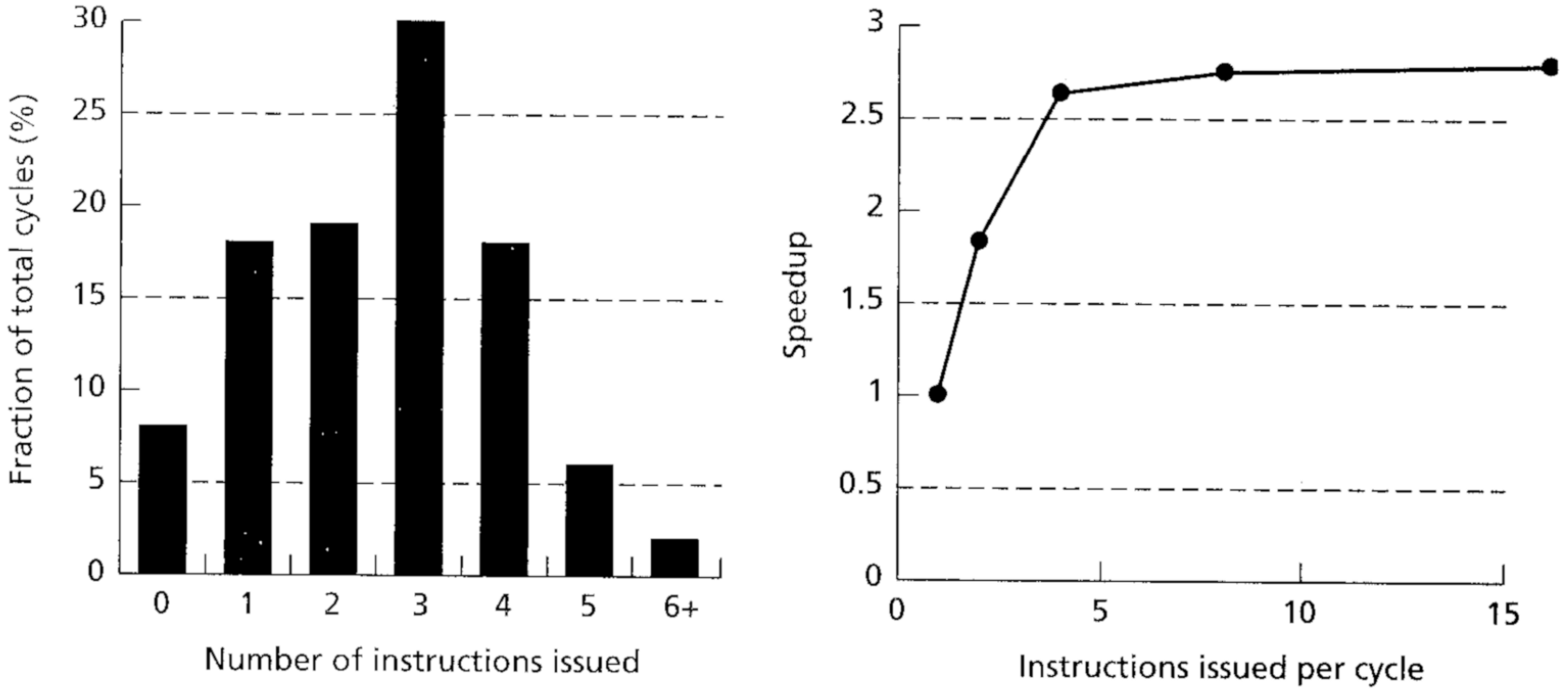


FIGURE 1.7 Distribution of potential instruction-level parallelism and estimated speedup under ideal superscalar execution. The figure shows the distribution of available instruction-level parallelism and maximum potential speedup under idealized superscalar execution, including unbounded processing resources and perfect branch prediction. Data is an average of that presented for several benchmarks by Johnson (1991).

Microprocessadores e parelelismo

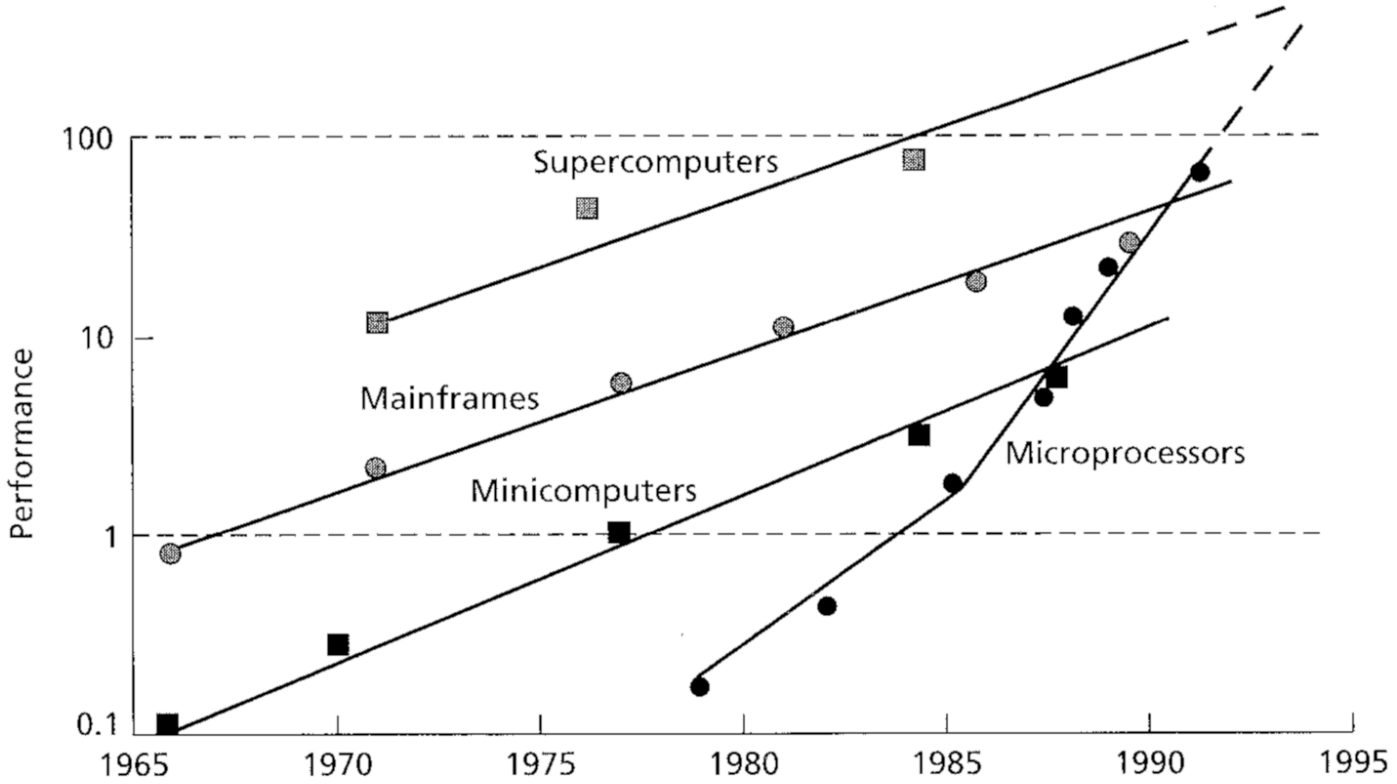


FIGURE 1.1 Performance trends over time of micros, minicomputers, mainframes, and supercomputers. Performance of microprocessors has been increasing at a rate of about 50% per year since the mid-1980s. More traditional mainframe and supercomputer performance has been increasing at a rate of roughly 25% per year. As a result, we are seeing the processor that is best suited to parallel architecture become the performance leader as well. *Source:* Hennessy and Jouppi (1991).

Diferentes paralelismos

- Sistemas computacionais têm:
  - Processador
  - Memória
  - Interconexão
- Qualquer deles pode ser paralelizado.
- Diferentes aplicações:
  - Intensivas em dados: precisam alta largura de banda no acesso a dados.
  - Servidores: precisam alta largura de banda de rede.
  - Aplicações científicas: precisam alta capacidade de processamento.
- Necessidades das aplicações precisam ser levadas em consideração.

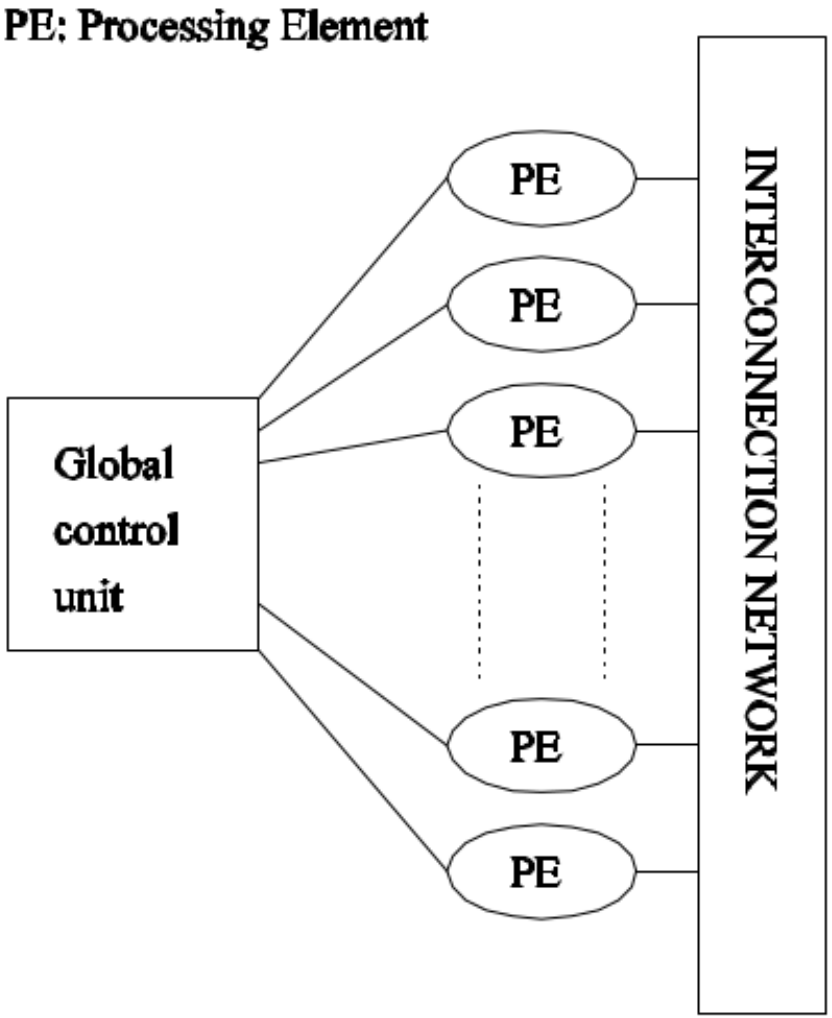
Programa paralelo

- Um programa paralelo precisa especificar:
  - Tarefas a serem executadas em paralelo (concorrência).
  - Comunicação e sincronização entre essas tarefas.
- Diversos níveis de granulosidade podem ser usados, desde instruções individuais a processos inteiros.
- A arquitetura paralela deve suportar o modelo utilizado.

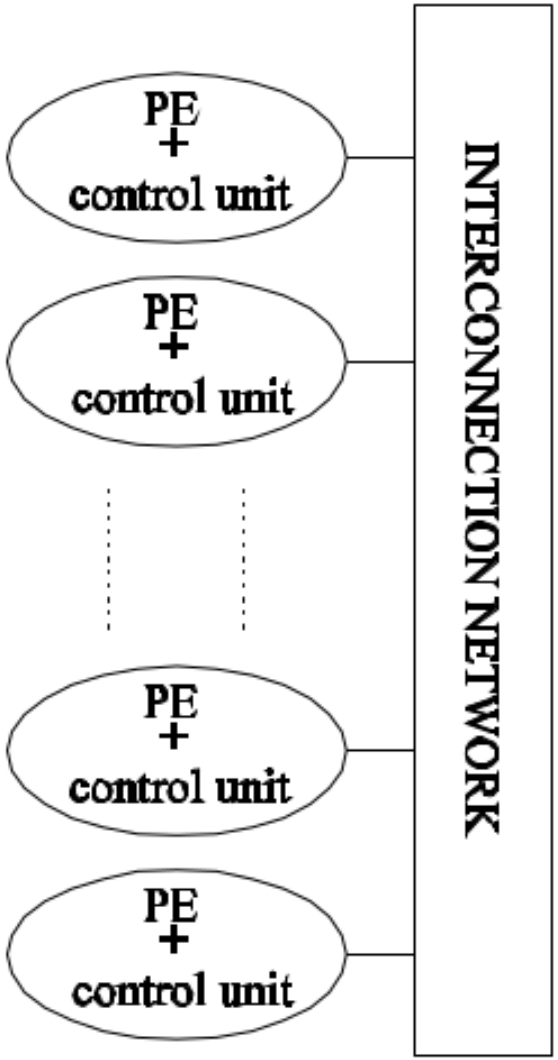
Estruturas de controle

- Unidades de processamento podem estar sobre controle centralizado (uma única unidade de controle) ou distribuído (uma unidade de controle por unidade de processamento).
- No primeiro caso (SIMD) a mesma instrução vai para todas as unidades de processamento, que operam sobre dados distintos. (SIMD: Single Instruction stream Multiple Data stream).
- No segundo caso (MIMD) cada processador executa seu próprio conjunto de instruções. (MIMD: Multiple Instruction stream Multiple Data stream).

SIMD


PE: Processing Element
PE
PE
PE
PE
PE
Global
control
unit
INTERCONNECTION NETWORK

MIMD


PE
+
control unit
PE
+
control unit
PE
+
control unit
PE
+
control unit
INTERCONNECTION NETWORK

SIMD x MIMD

- SIMD:
  - Precisa estruturas regulares de computação (exemplo: imagens, áudio).
  - Execuções condicionais implicam deixar alguns processadores parados ocasionalmente.
  - Requerem menos hardware, mas o hardware é especialmente projetado.
  - Exemplo: instruções MMX, SSE, etc.
- MIMD:
  - Para muitos processadores, precisa executar o mesmo programa. (SPMD: Single Program Multiple Data).
  - Processadores podem ser CPU comerciais normais.
  - Requerem mais hardware, mas pode ser hardware comercial (produzido em escala).
  - Exemplos: processadores multicore e clusters.

Modelos de comunicação

Dois modelos de troca de dados entre processos paralelos:

- Acesso a dados compartilhados.
- Troca de mensagens.

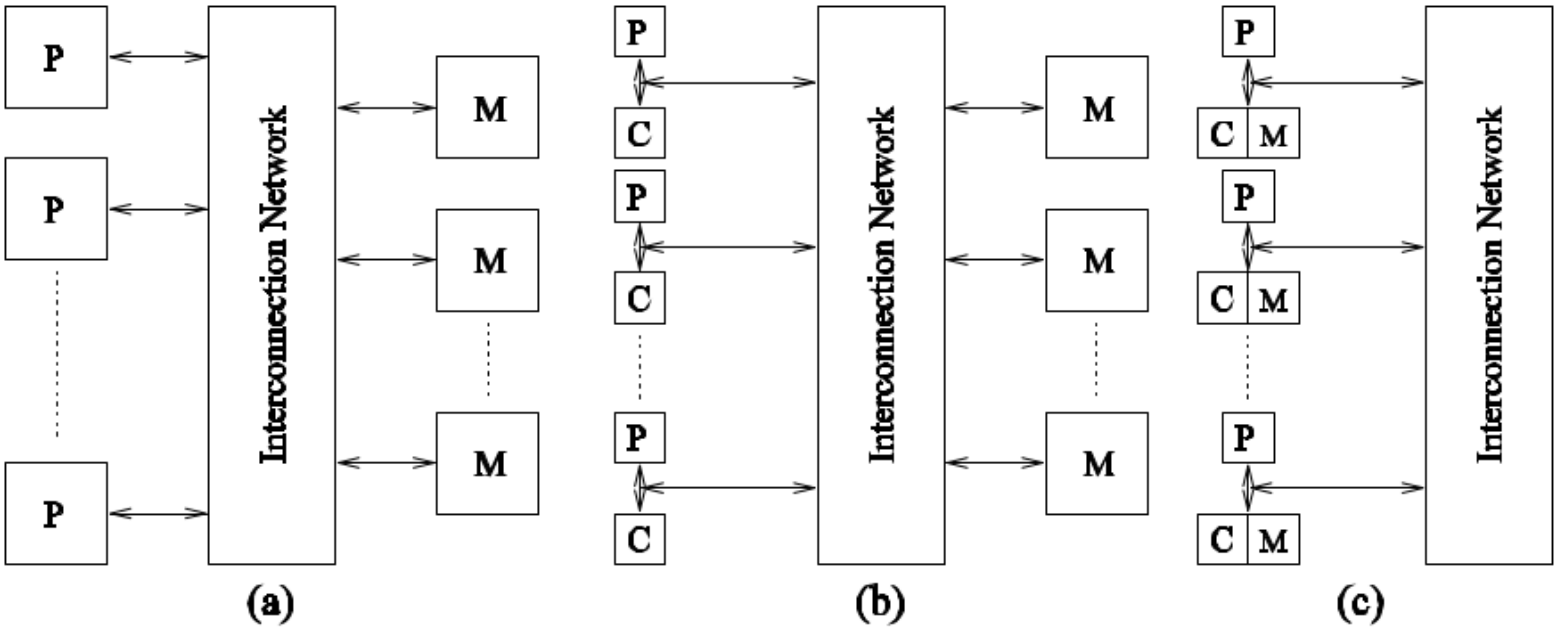
Arquiteturas correspondentes:

- Máquinas de espaço de endereçamento compartilhado (multiprocessadores).
- Máquinas de passagem de mensagens (multicomputadores).

Memória comparilhada

- Parte ou toda a memória é compartilhada (acessível a todos os processadores).
- Interação entre processadores ocorre por escritas e leituras em memória compartilhada.
- Se o tempo de acesso de todas as posições de memória é o mesmo, temos UMA (Uniform Memory Access); se não, temos NUMA (Non Uniform Memory Access).

UMA e NUMA


P
P
P
Interconnection Network
M
M
M
(a)
P
C
P
C
P
C
Interconnection Network
M
M
M
(b)
P
C M
P
C M
P
C M
Interconnection Network
(c)

UMA x NUMA

- Importante ao escrever algoritmos pois em NUMA há um nível a mais de localidade (acesso a bloco de memória local é mais rápido).
- Em geral, NUMA mais eficiente (alguns dos acessos não precisam passar pela rede de interconexão).
- Caches precisam ser coordenados (ver a seguir).

Passagem de mensagens

- Cada processador tem sua própria memória.
- Comunicação ocorre por envio e recepção de mensagens.
- Operações de comunicação são operações de entrada e saída.

Redes de interconexão

- Carregam dados de um processador para outro ou entre processadores e memórias.
- Feitas de:

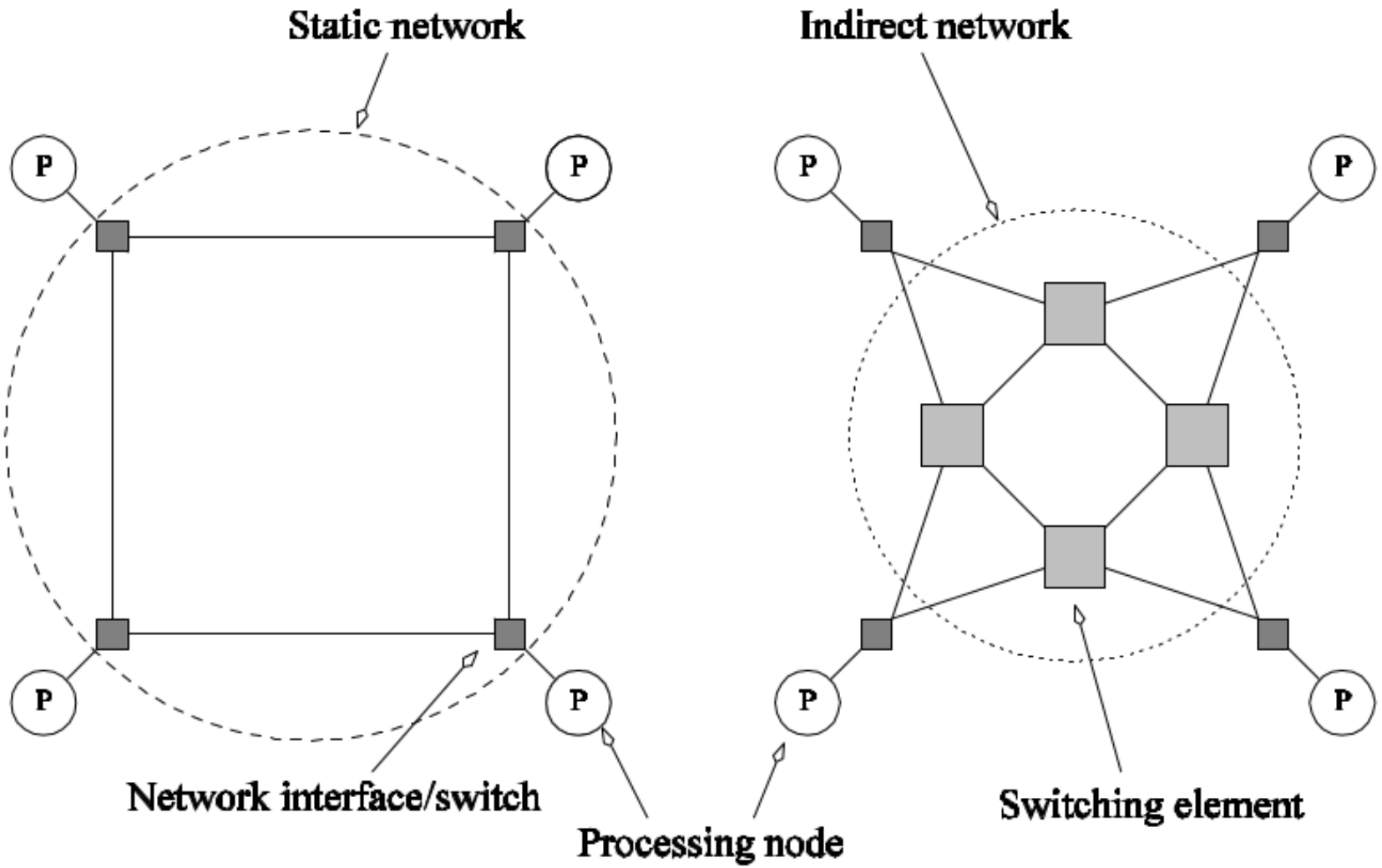
  elos (links) Fios ou fibras

  chaves (switches) permitem a ligação de elos específicos.
- Podem ser:

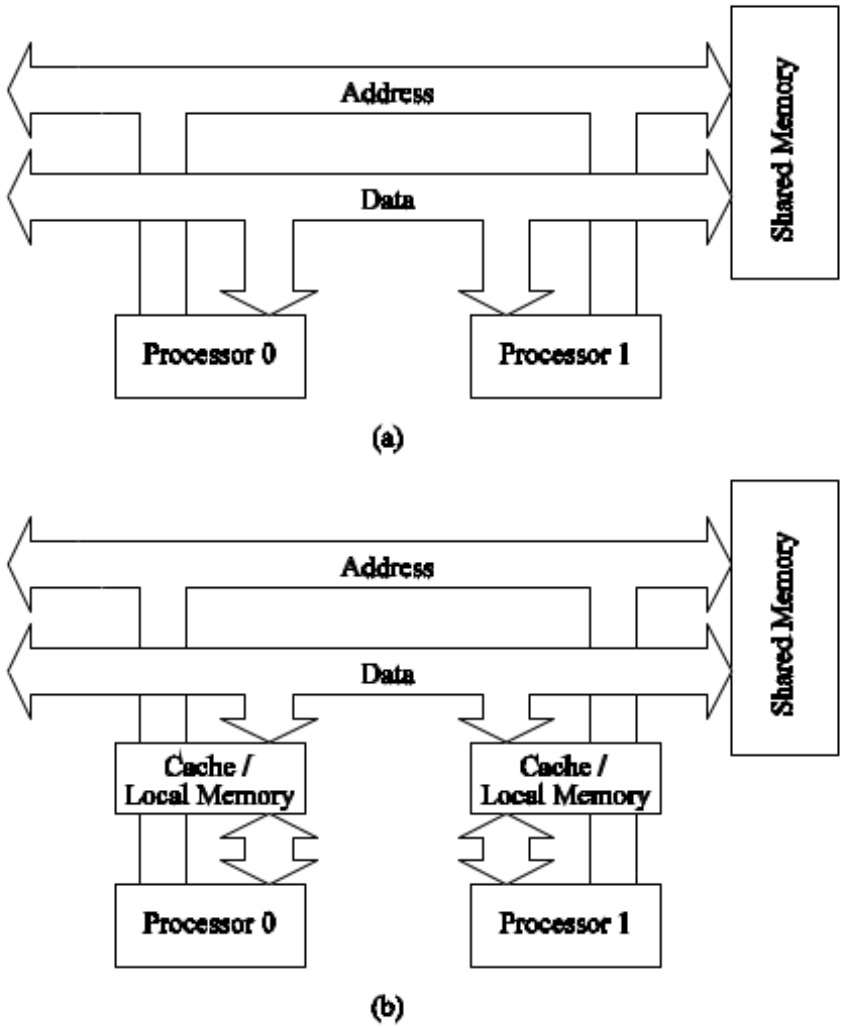
  estáticas (diretas) elos ponto a ponto entre dois elementos de processamento.

  dinâmicas (indiretas) elos interligando chaves e elementos de processamento.

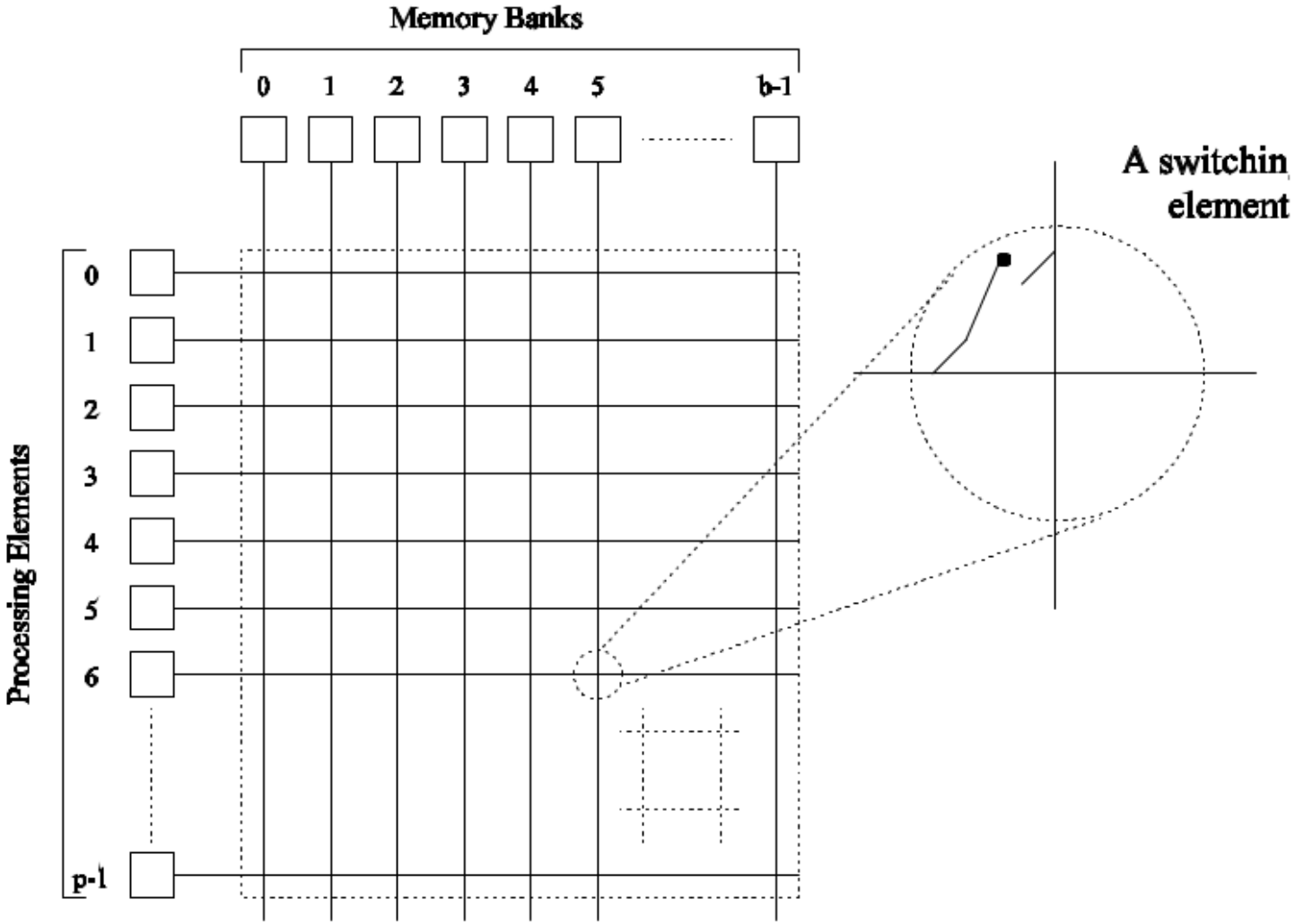
Estáticas e dinâmicas



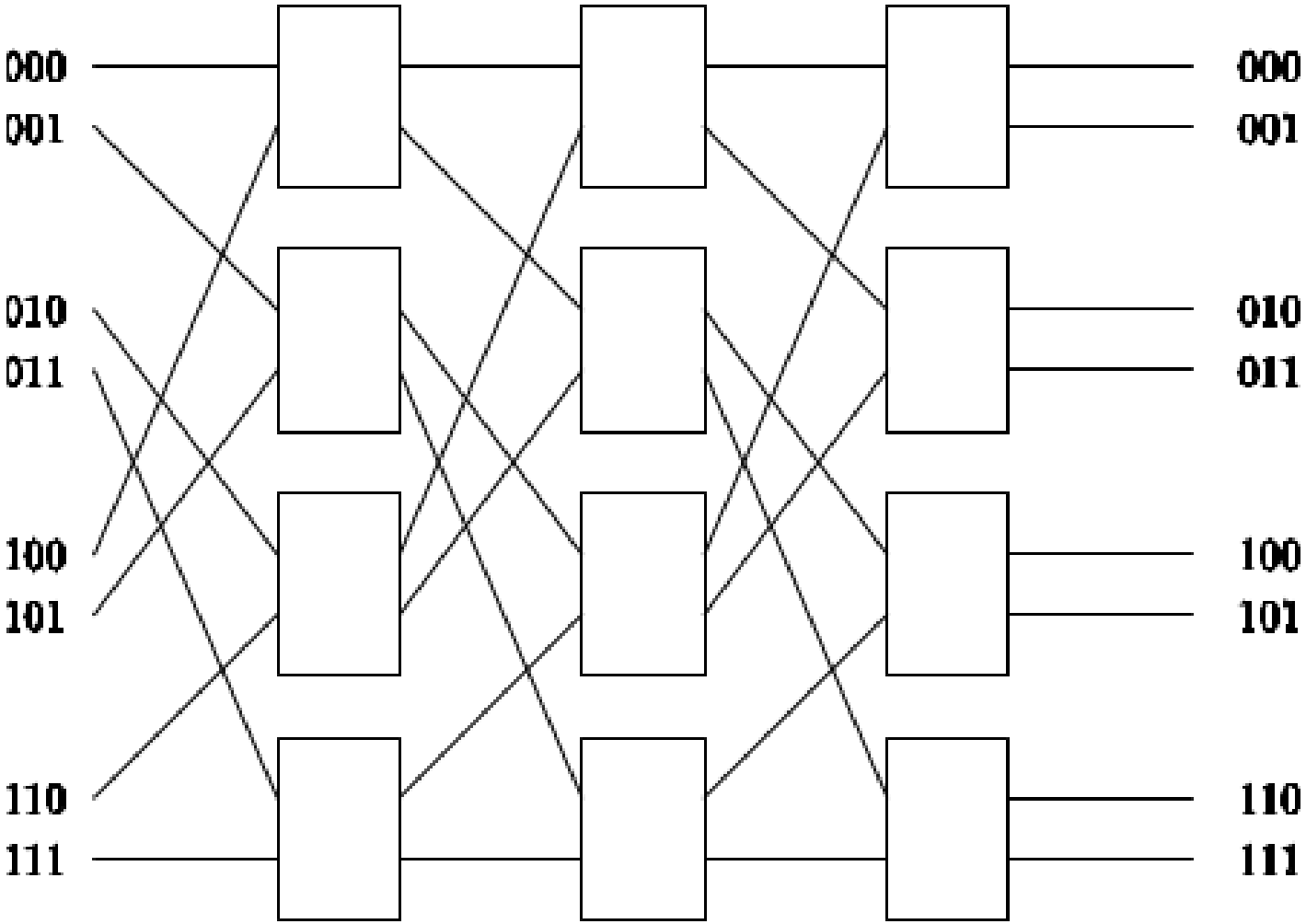
Redes dinâmicas: Barramentos



Redes dinâmicas: Crossbar



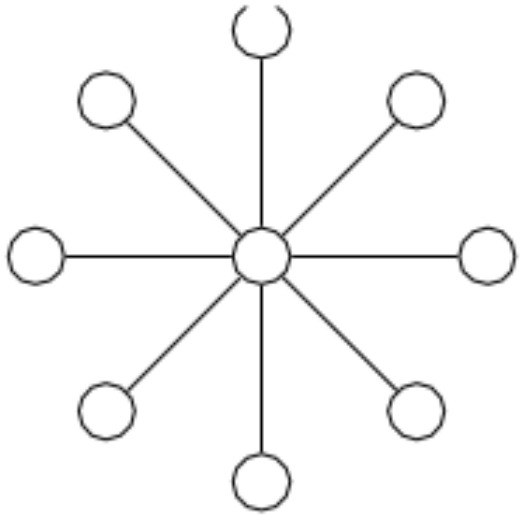
Redes dinâmicas: Omega



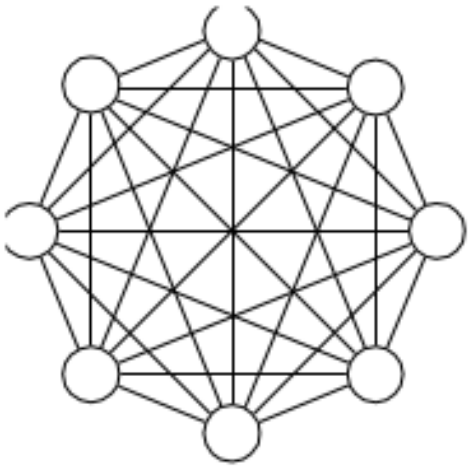
Redes dinâmicas: Comparação

- Barramentos são baratos, mas largura de banda é limitada (não cresce com o número de processadores p).
- Crossbar tem largura de banda que cresce com p, mas é muito cara (custo cresce com p^2).
- Redes multiestágio (exemplo: omega), são um compromisso: largura de banda cresce com o número de processadores e custo não cresce tanto quanto crossbar (cresce com $p \log p$).

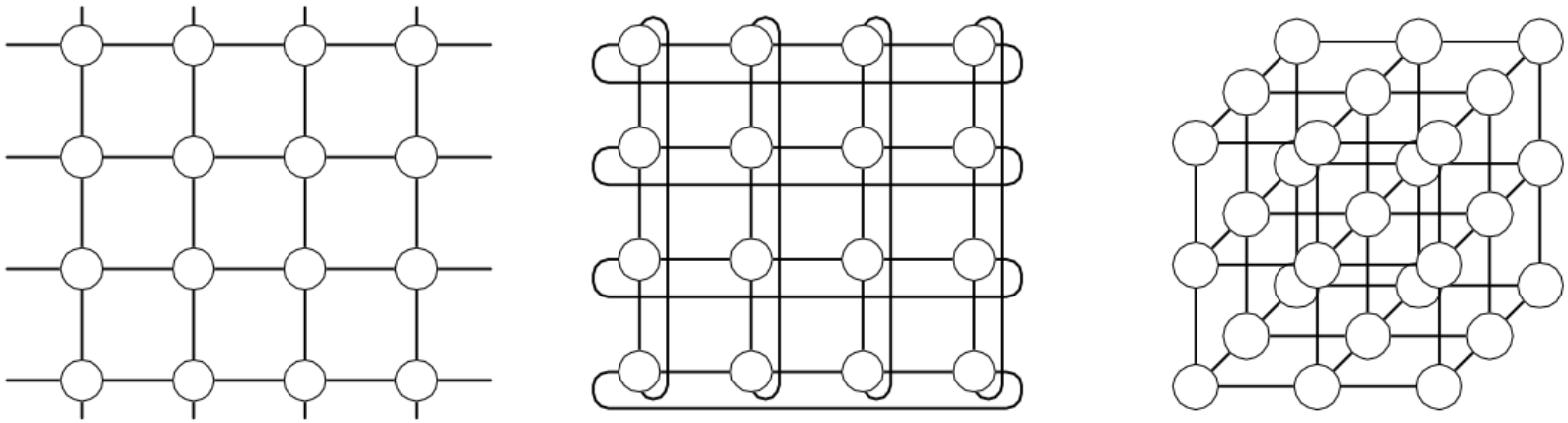
Redes estáticas: Estrela



Redes estáticas: Todos com todos

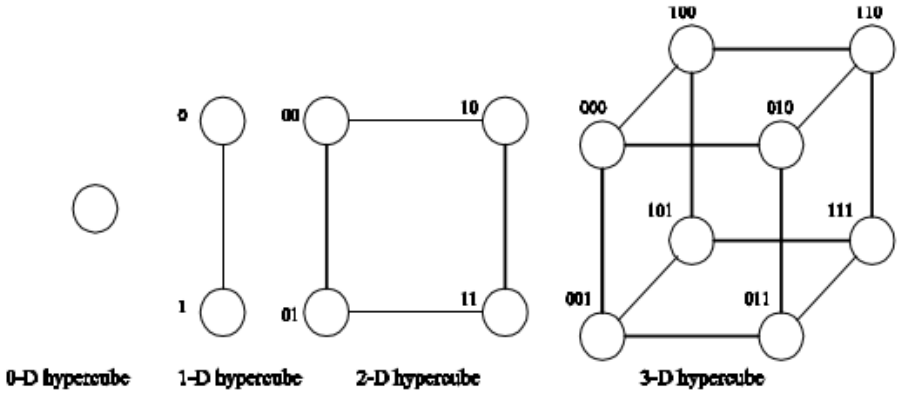


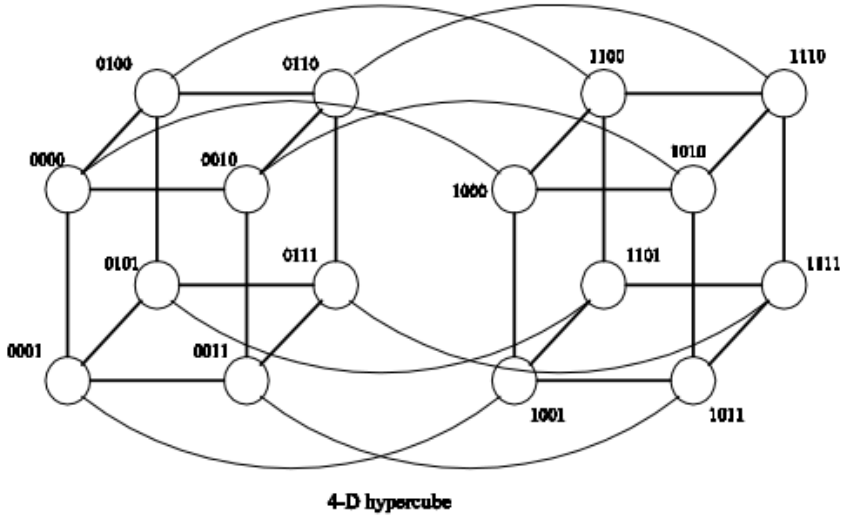
Redes estáticas: Malhas



(a) (b) (c)

Redes estáticas: Hipercubo





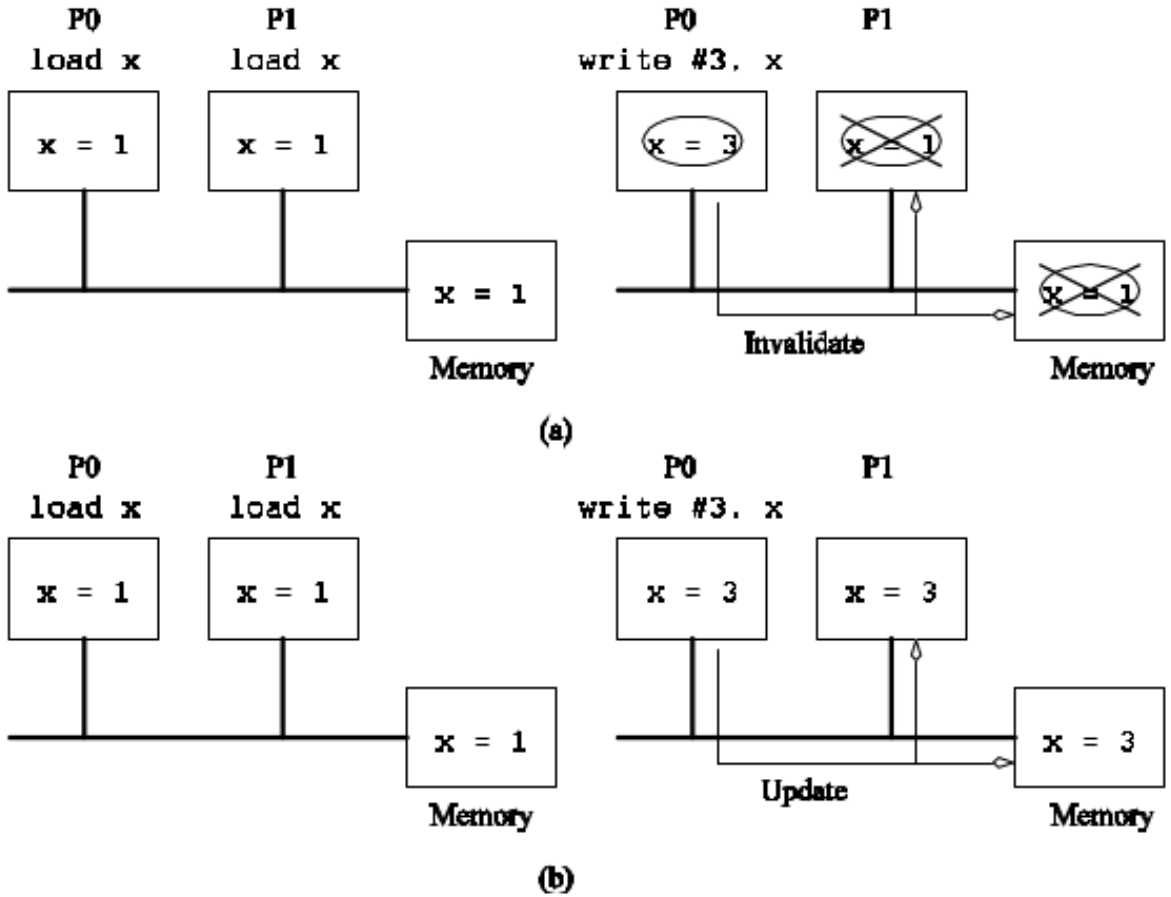
Redes estáticas: Comparação

- Estrelas são baratas (apenas um elo para cada novo processador), mas não são escaláveis (todas as mensagems passam pelo nó central).
- Todos com todos é escalável, mas é muito cara (número de elos cresce com p^2).
- Malhas e hipercubo são compromissos.
- Malha: custo cresce proporcionalmente ao número de processadores (número de elos por processador é constante).
- Hipercubos: custo cresce mais rapidamente que o número de processadores (número de elos por processador cresce com $\log p$).
- Hipercubos são bem adaptados para diversos algoritmos paralelos.
- Existe um grande número de outras topologias.

Coerência de cache

- Quando há caches em sistemas de memória compartilhada, podem existir várias cópias da mesma posição de memória.
- O sistema de caches precisa assegurar que os acessos fazem sentido (por exemplo, não ler valor desatualizado).
- Isto é conhecido como *coerência de cache*.

Update e invalidate



False sharing

Quando dois elementos distintos na memória são acessados por diferentes processadores (e portanto não há compartilhamento entre os processadores), mas os dois elementos ficam *na mesma linha de cache*, dizemos que temos false sharing.
False sharing pode ocasionar alto custo no acesso à memória, e deve ser evitado (por exemplo, não distribuindo elementos consecutivos de um vetor para processadores diferentes).

Implementando protocolo de invalidação

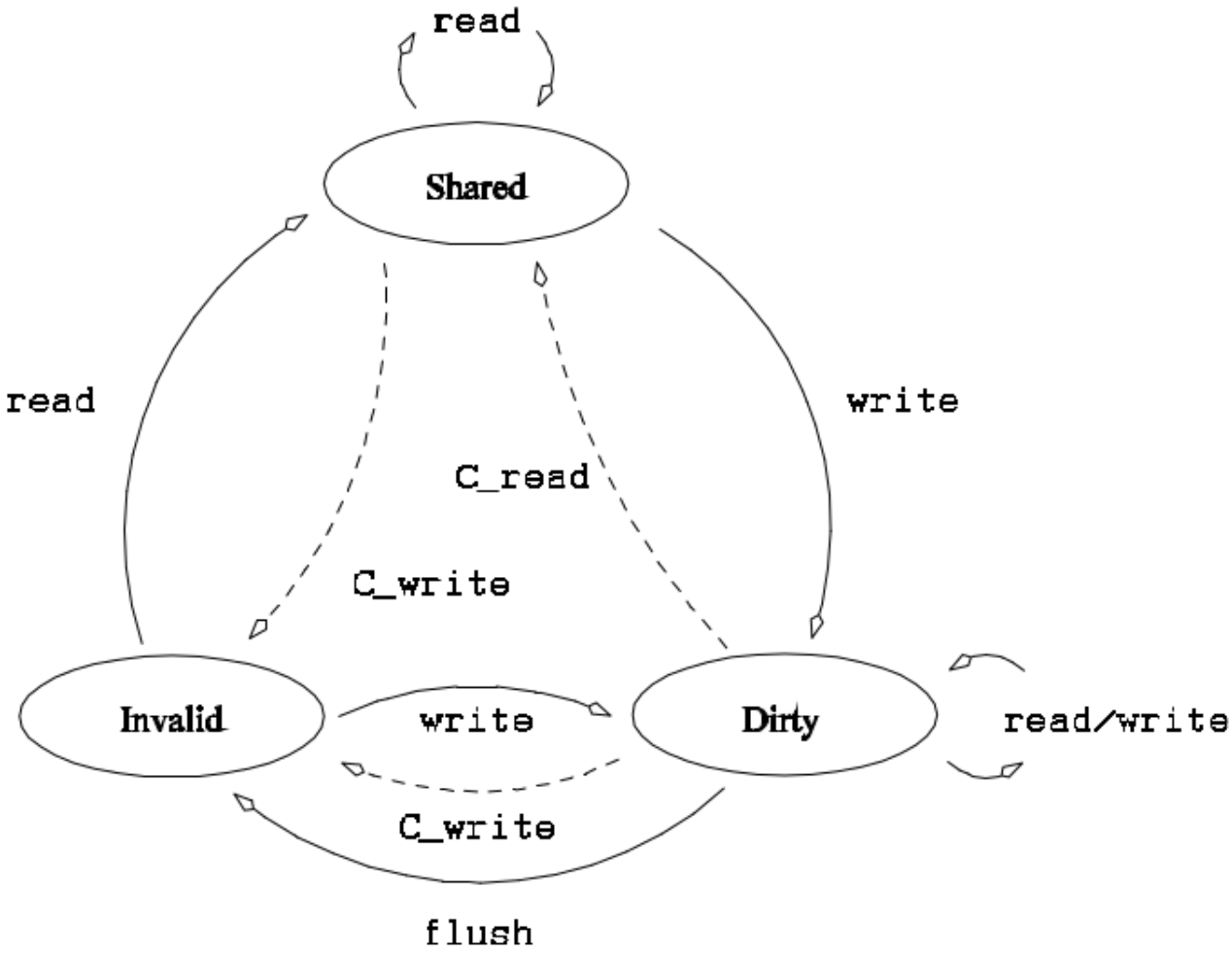
- Cada bloco de memória é associado com um estado:

  Compartilhado Existem múltiplas cópias válidas do bloco.

  Inválido A cópia presente na linha local é inválida (valor foi atualizado em outro processador).

  Sujo Apenas uma cópia existe do bloco.

Protocolo de invalidação



Exemplo

Time ↓

Instruction at Processor 0	Instruction at Processor 1	Variables and their states at Processor 0	Variables and their states at Processor 1	Variables and their states in Global mem.
				x = 5, D y = 12, D
read x		x = 5, S		x = 5, S
	read y		y = 12, S	y = 12, S
x = x + 1		x = 6, D		x = 5, I
	y = y + 1		y = 13, D	y = 12, I
read y		y = 13, S	y = 13, S	y = 13, S
	read x	x = 6, S	x = 6, S	x = 6, S
x = x + y		x = 19, D	x = 6, I	x = 6, I
	y = x + y	y = 13, I	y = 19, D	y = 13, I
x = x + 1		x = 20, D		x = 6, I
	y = y + 1		y = 20, D	y = 13, I

Implementação do protocolo

- Como transmitir as sinais do protocolo para os interessados:
- Duas formas:

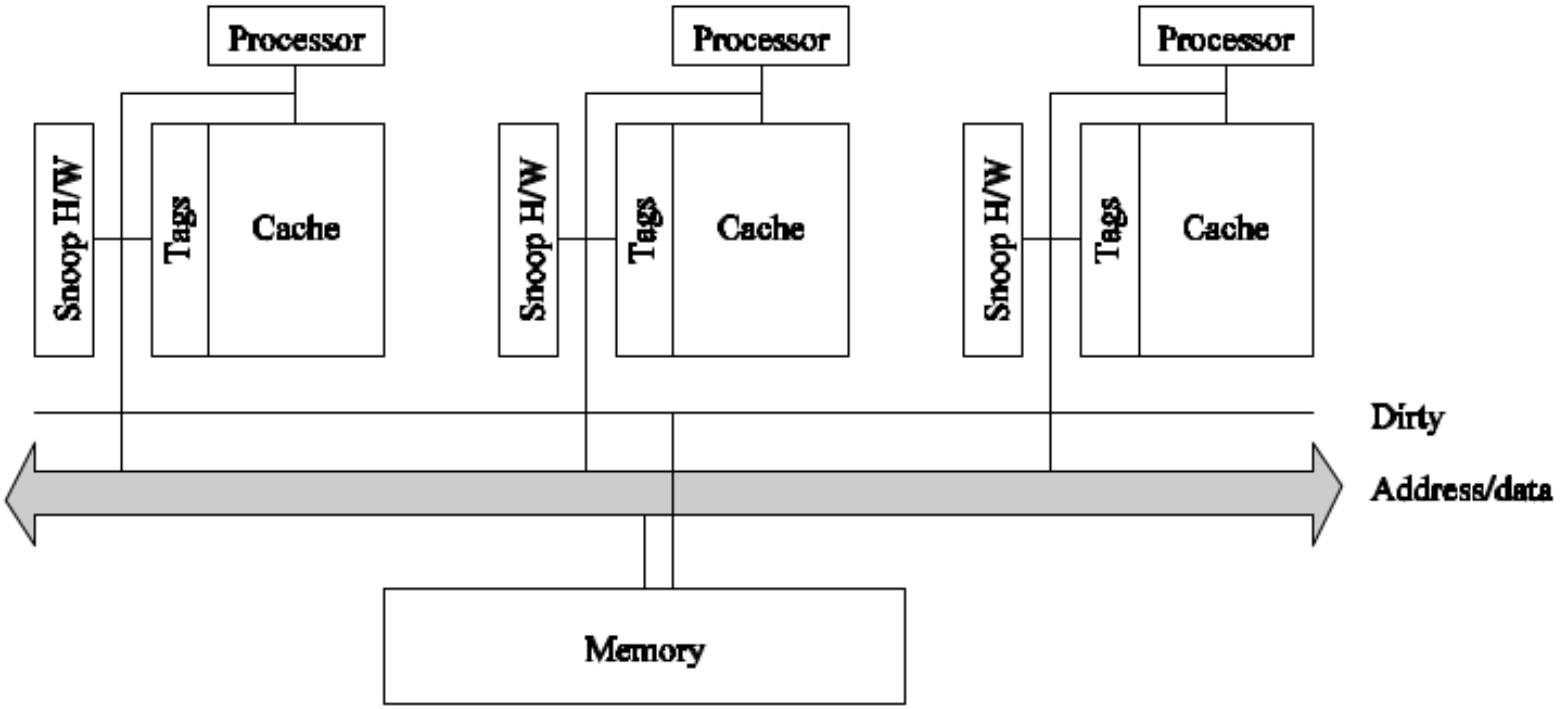
  Snooping Usa um barramento.

  Diretory Usa uma lista de interessados.

Snooping

- Um barramento ou outro meio que permite transmissão de mensagens para todos os processadores existe.
- Os sinais de transição de estado são enviados por esse meio.
- Todos os processadores ficam escutando esse meio e detecam os sinais quando enviados.
- Se o sinal for relativo a um bloco guardado localmente, toma a atitude necessária.

Snooping


Processor
Processor
Processor
Snoop H/W
Tags
Cache
Snoop H/W
Tags
Cache
Snoop H/W
Tags
Cache
Dirty
Address/data
Memory

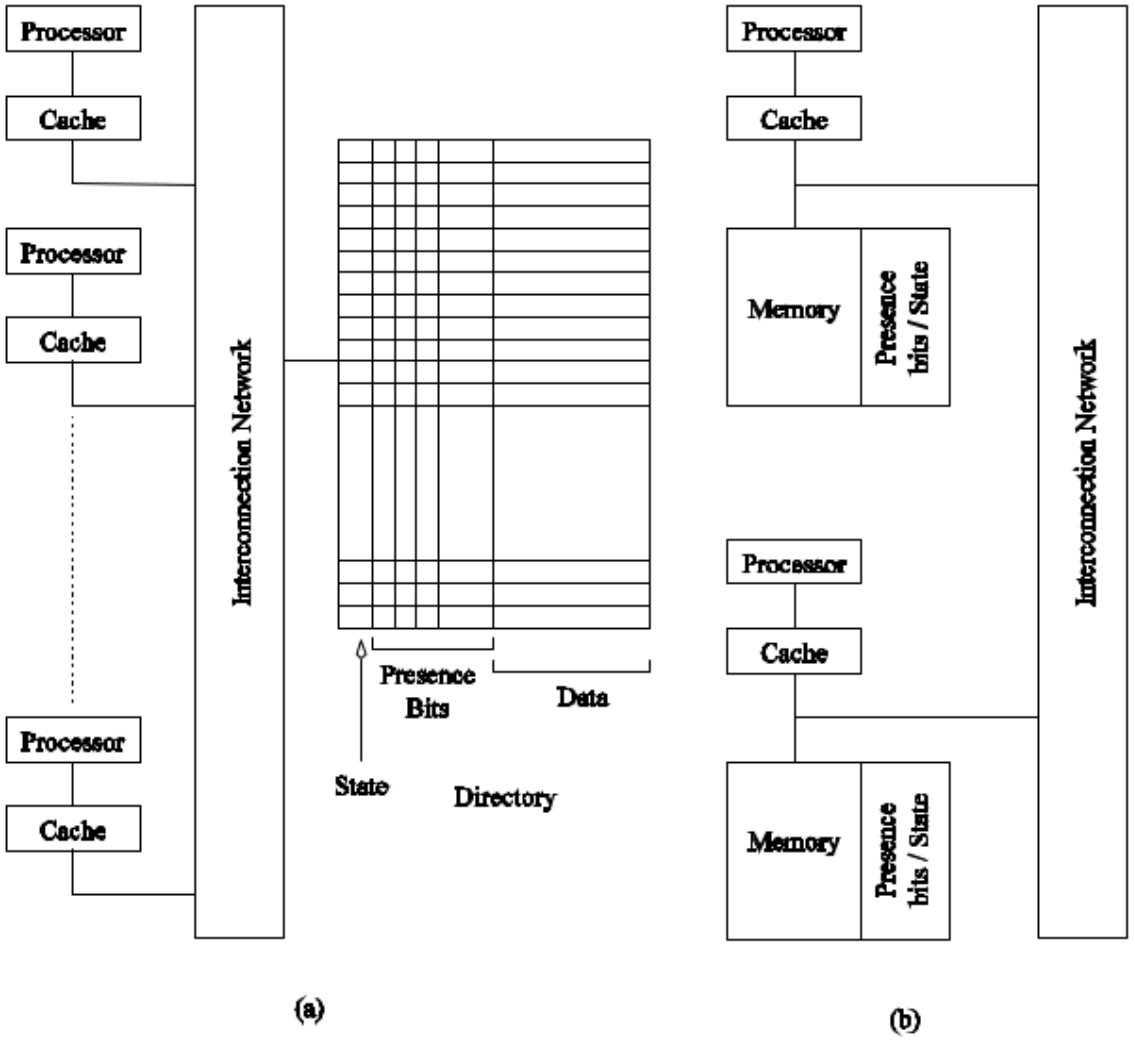
Desempenho de snooping

- Uma linha marcada *suja* pode ser acessada localmente (leitura e escrita) sem gerar tráfego no meio de comunicação.
- Um dado apenas lido por vários processadores é marcado como *compartilhado*, e todas as leituras posteriores não geram tráfego.
- Dados que são lidos e escritos por vários processadores geram tráfego de coerência, que se adiciona aos tráfegos de leitura e escrita na memória.

Directory

- No esquema de snooping, a necessidade de envio de todo o tráfego de coerência para todos os processadores limita a escalabilidade para grande número de processadores.
- No esquema de diretório, as informações de coerência são enviadas apenas para os processadores interessados.
- Para cada linha, precisamos manter a informação de todos os processadores que têm uma cópia do bloco de memória guardado nela.

Directory


Processor
Cache
Processor
Cache
Processor
Cache
Interconnection Network
Presence Bits
Data
State
Directory
(a)
Processor
Cache
Memory
Presence bits / State
Processor
Cache
Memory
Presence bits / State
Interconnection Network
(b)

Desempenho de directory

- Diminuimos as comunicações de operações de coerência no meio de comunicação.
- Precisamos dos bits de presença. Para um grande número de processadores, a sobrecarga pode ser grande.
- O diretório é um gargalo, portanto soluções de diretório distribuído são necessárias.

Custo de comunicação

- Custos de comunicação são importantes em programas paralelos.
- Eles depende de uma grande número de fatores.
- Vamos usar um modelo simplificado (linear) com dois parâmetros:

  Tempo de partida (startup time) Tempo para a mensagem sair do remetende e começar a chegar no destinatário. Representado por t_s

  Tempo por palavra (per word transfer time) Tempo gasto adicionalmente para a transmissão de cada uma das palavras da mensagem. Representado por t_w.
- O tempo de comunicação de n palavras será então dado por:

$$t(n) = t_s + nt_w.$$