Organização de Computadores II
DCC007

Aula 17 – Paralelismo em Nível de Thread

Prof. Omar Paranaiba Vilela Neto

Motivações para Arquiteturas Paralelas

- Desempenho
- Custo
- Disponibilidade e confiabilidade

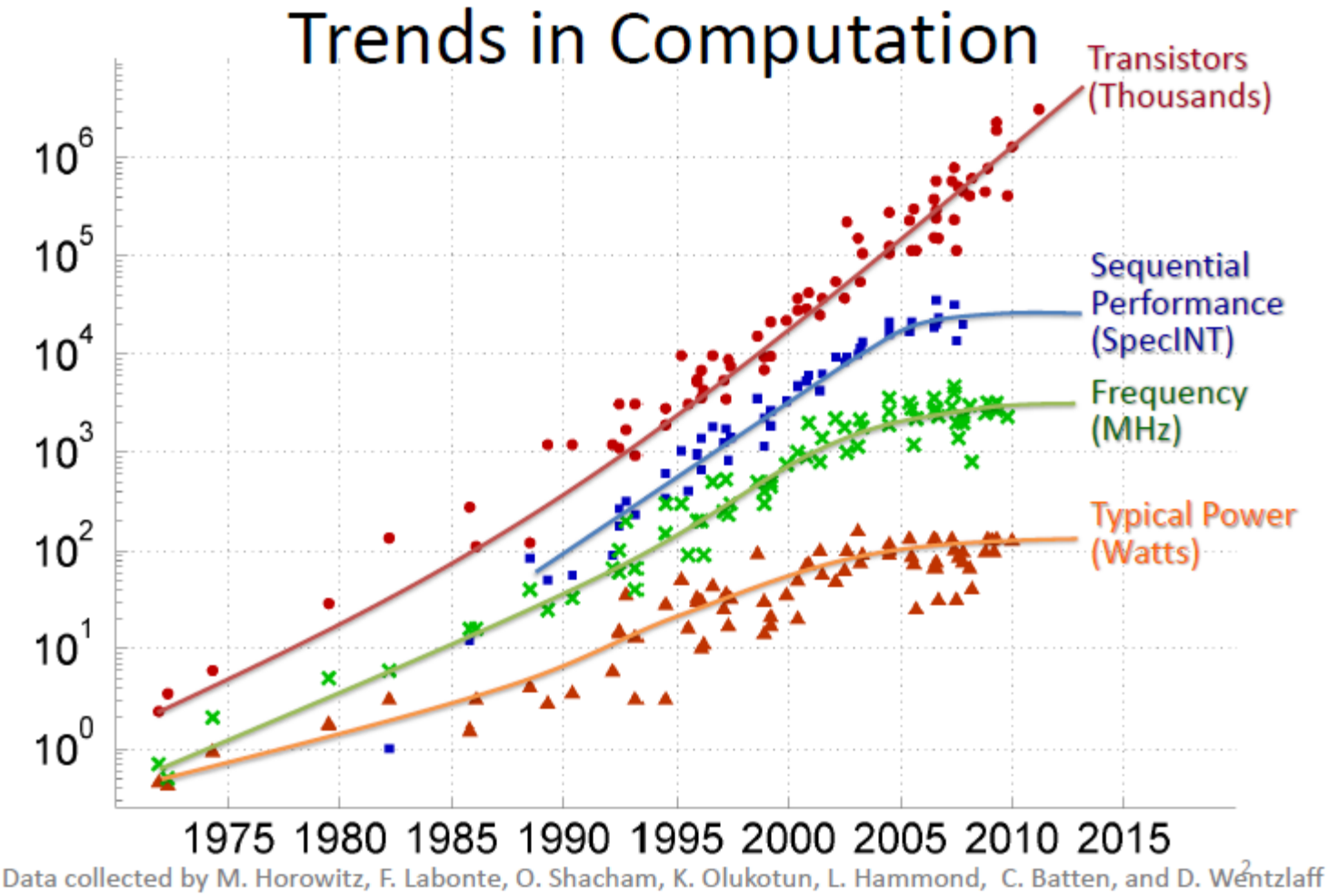
Arquitetura Paralela

- Máquina Paralela: conjunto de elementos de processamento (PEs) que cooperam para resolver problemas computacionalmente difíceis rapidamente
  - Quantos PEs?
  - Poder computacional de cada PE?
  - Quanta memória em cada PE?
  - Como o dado é transmitido entre PEs?
  - Quais as primitivas para cooperação?
  - Qual o desempenho?
  - Como a máquina escala?

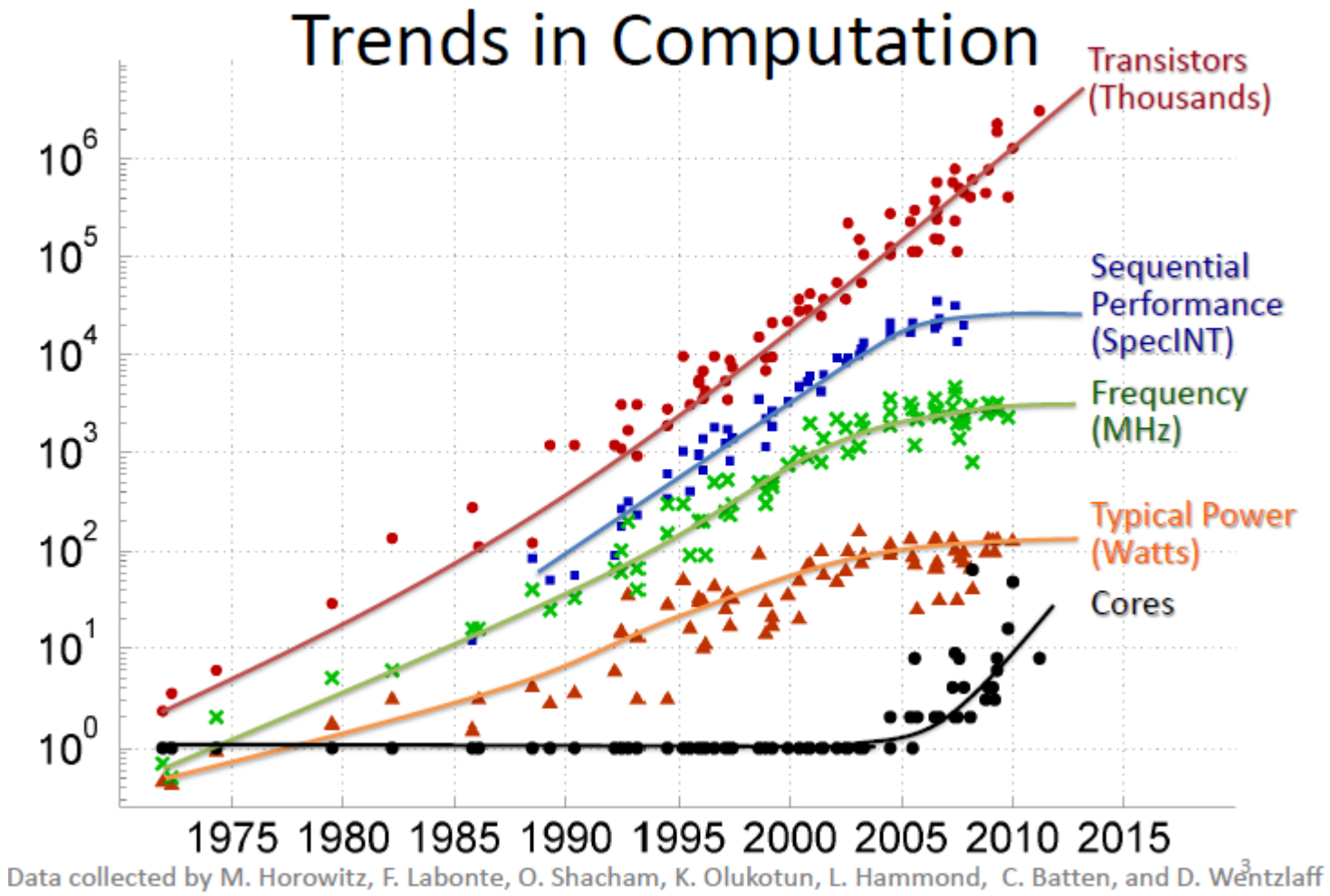
Por que Estudar Arquiteturas Paralelas

- Alternativa para clocks mais rápidos para melhorar desempenho
  - já vimos isso também em processadores superescalares
- Aplica-se em todos os níveis do projeto de sistemas
- Abre novas perspectivas para **arquitetura de computadores**
  - Por que não colocar múltiplos processadores em um um único chip ao invés de colocar um processador superescalar?

Por que Estudar Arquiteturas Paralelas



Por que Estudar Arquiteturas Paralelas



Arquiteturas Paralelas

- Demanda das aplicações: progresso contínuo de software paralelo
  - Explorar ao máximo o tempo de CPU
  - Computação científica: Biologia, Química, Física, **Nanotecnologia**
  - Computação de uso geral: Vídeo, Computação Gráfica, CAD, Banco de Dados, **Big Data**

- Tendências da tecnologia e das arquiteturas
  - Tecnologia
    - # de transistores crescendo **rapidamente**
    - Frequência crescendo **moderadamente**
  - Arquitetura
    - Limites de ILP

- Paralelismo de granularidade grossa, mais viável

Tendências Arquiteturais ILP

- Speedups reportados para processadores superescalares
  - Horst, Harris, Jardine [1990] 1,37
  - Wang, Wu [1988] 1,70
  - Smith, Johnson, Horowitz [1989] 2,30
  - Murakami, ... [1989] 2,55
  - Jouppi, Wall [1989] 3,20
  - Lee, Kwok, Briggs [1991] 3,50
  - Melvin, Patt [1991] 8,00
  - Butler, ... [1991] 17+

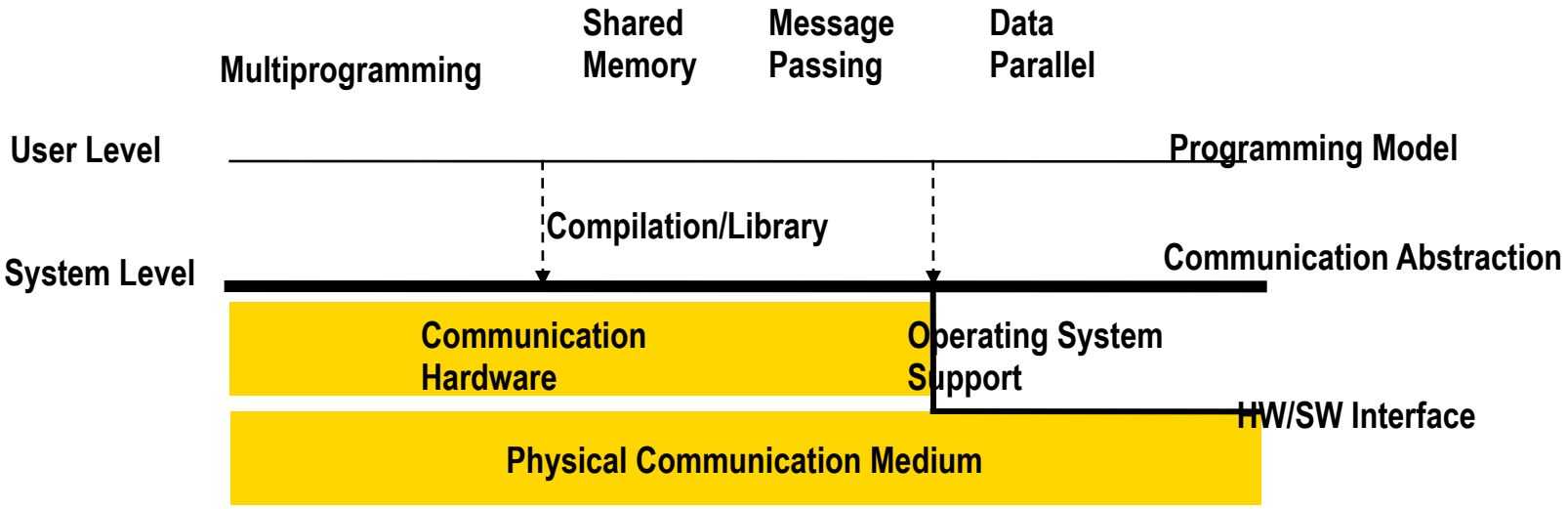
- Melhorias em ILP continuarão a prevalecer nos próximos anos?

Arquiteturas Paralelas e Computação Científica

- Computação Científica
  - uPs conquistaram ganhos altos em desempenho de FP
    - clocks
    - FPUs com pipeline (mult-add todo ciclo)
    - Uso efetivo de caches
  - uPs são relativamente baratos
    - Custo de desenvolvimento de dezenas de milhões de dólares amortizado sobre os milhões de componentes vendidos
- Multiprocessadores usando uPs de grande-escala **substituíram** supercomputadores tradicionais

Arquiteturas Paralelas Hoje

- Extensão de arquitetura de computadores para suportar comunicação e cooperação
  - ANTIGAMENTE: Instruction Set Architecture
  - HOJE: Arquitetura de comunicação



Arquitetura de Comunicação

= Abstração de Comunicação + Implementação

- Abstração:
  - Primitivas de comunicação de HW/SW para o programador (ISA)
  - Modelo de memória compartilhada, baseado em mensagens, ...
  - Primitivas devem ser eficientemente implementadas
- Implementação
  - Onde interface de rede e controlador de comunicação integram no nó?
  - Rede de interconexão
- Objetivos:
  - Aplicações de uso geral (custo/aplicação)
  - Programabilidade
  - Escalabilidade

Considerações Sobre Escalabilidade

- Tanto máquinas small-scale quanto large-scale tem seu lugar no mercado

- Custo-desempenho-complexidade são diferentes para cada máquina

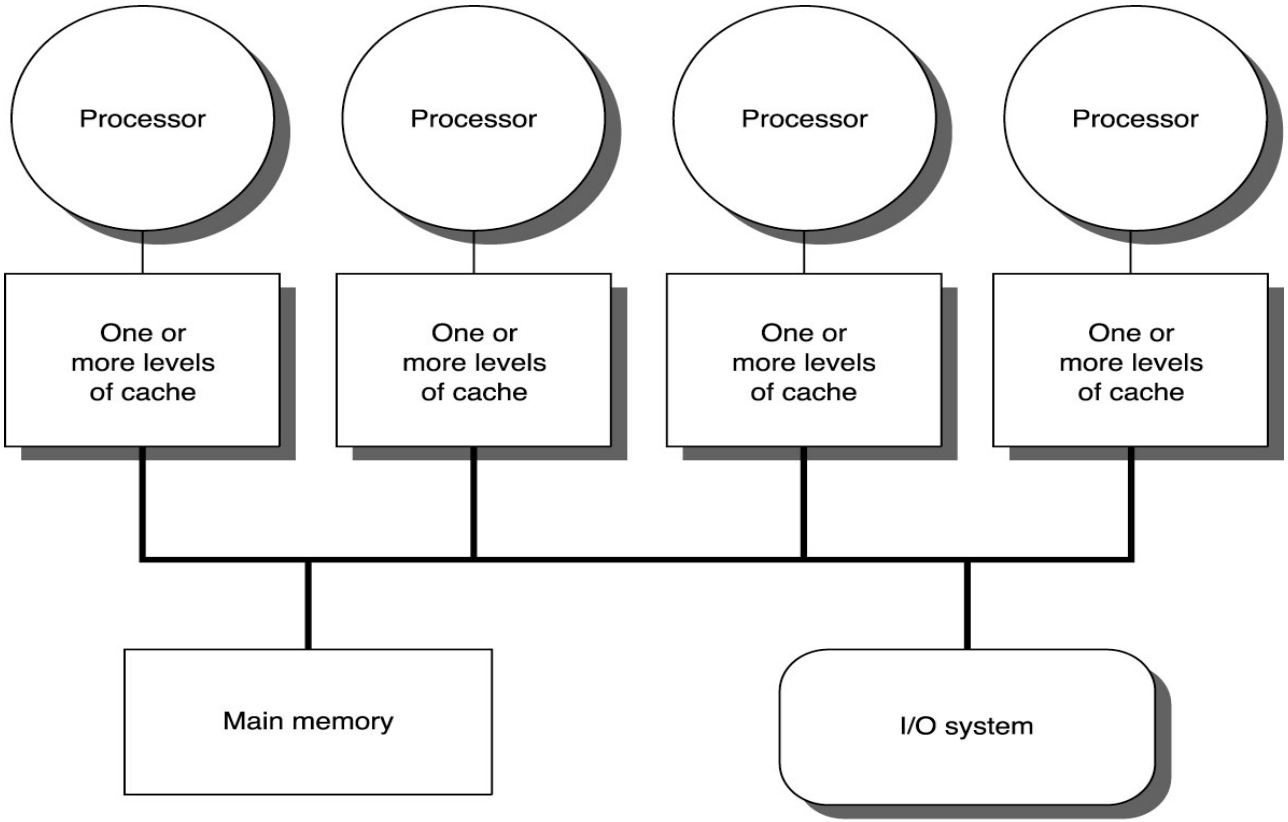
Questões de Projeto

- ***Espaço de endereço***: Como dados compartilhados e/ou comunicação são nomeados?
- ***Latência***: Qual é a latência da comunicação?
- ***Bandwidth***: Quanto dado pode ser comunicado por segundo?
- ***Sincronização***: Como a transferência de dados pode ser sincronizada?
- ***Granularidade***:

  Silício = processador + memória
- ***Aplicabilidade***: Propósito geral ou específico?

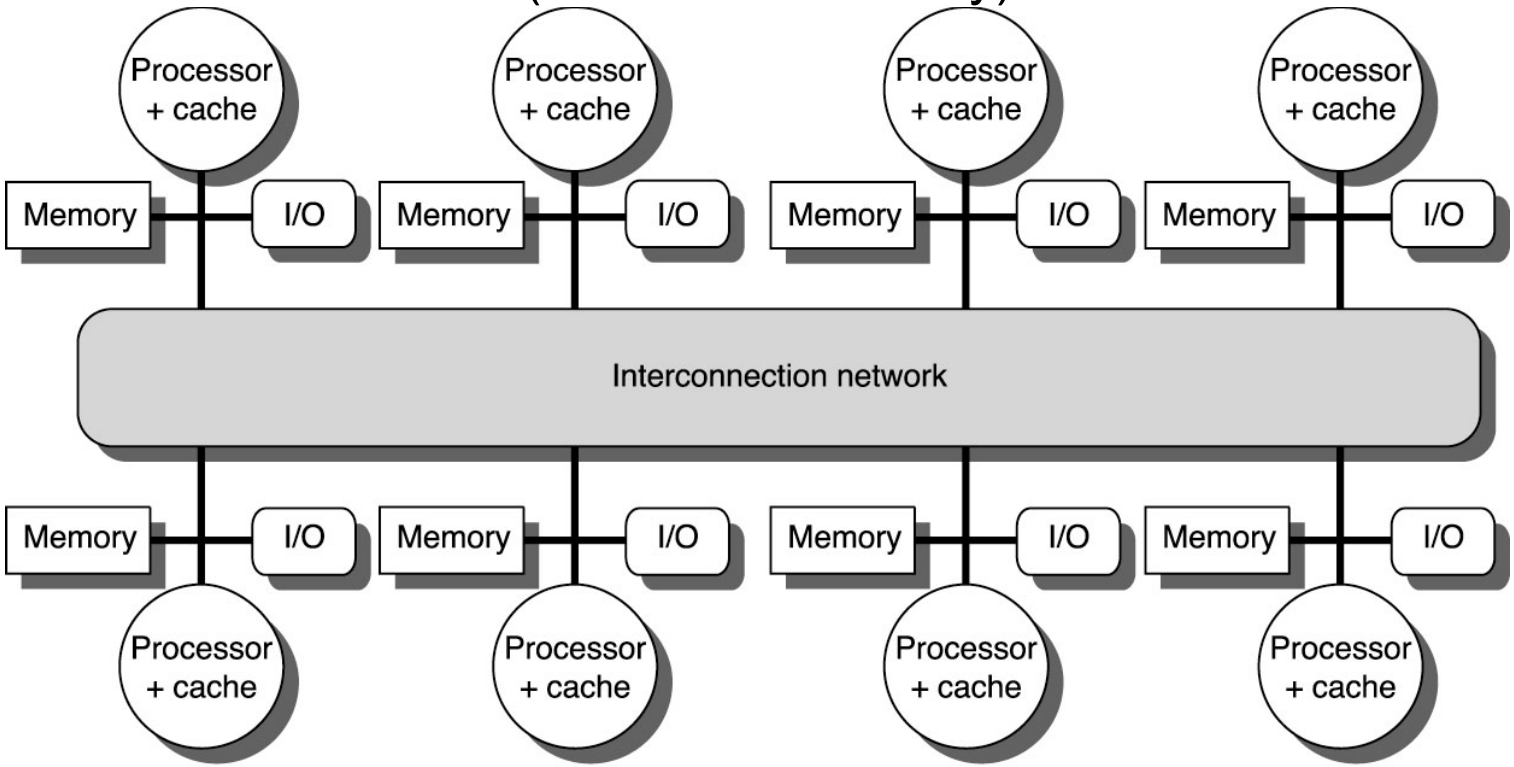
Small-Scale MIMD

- Memória: centralizada com uniform access time (“uma”) e conexão por barramento



Large-Scale MIMD

- Memória: distribuída com nonuniform access time (“numa”) interconexão escalável (distributed memory)

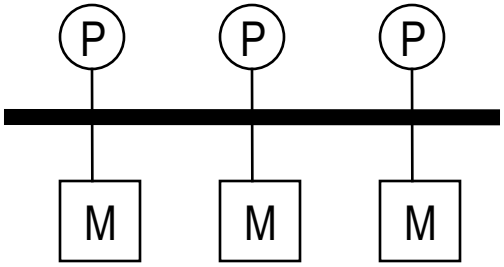


Modelo de Comunicação

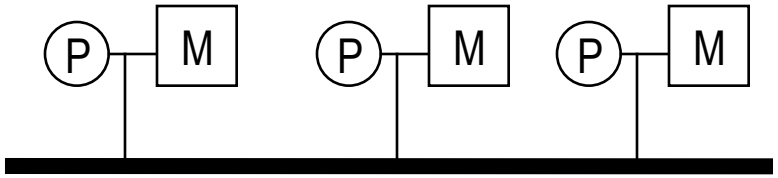
- ***Shared Memory*** (centralizada ou distribuída)
  - Processadores comunicam com espaço de endereçamento compartilhado
  - Fácil em máquinas small-scale
  - Vantagens:
    - Escolhido para uniprocessadores, MPs small-scale
    - Fácil de programar
    - Baixa latência
    - Mais fácil para usar hardware de controle de cache
- ***Message passing (RPC)***
  - Processadores possuem memórias privadas e comunicam-se via mensagens
  - Vantagens:
    - Menos hardware, fácil de projetar
    - Escalabilidade
- HW pode suportar os dois modelos

Arquiteturas com Espaço de Endereçamento Compartilhado

- Todos os processadores podem **acessar** todas as posições de memória do sistema, provendo um mecanismo conveniente e rápido para comunicação



Centralizada
(small-scale)



Distribuída
(large-scale)

- Conhecidas também como shared memory

Modelo de Compartilhamento de Memória

- Comunicação, compartilhamento e sincronização existem via load/store em variáveis compartilhadas
- Modelo de programação relativamente fácil (uniprocessador + sincronização)
- Desvantagem: potencial e escalabilidade

Arquiteturas Baseadas em Troca de Mensagens

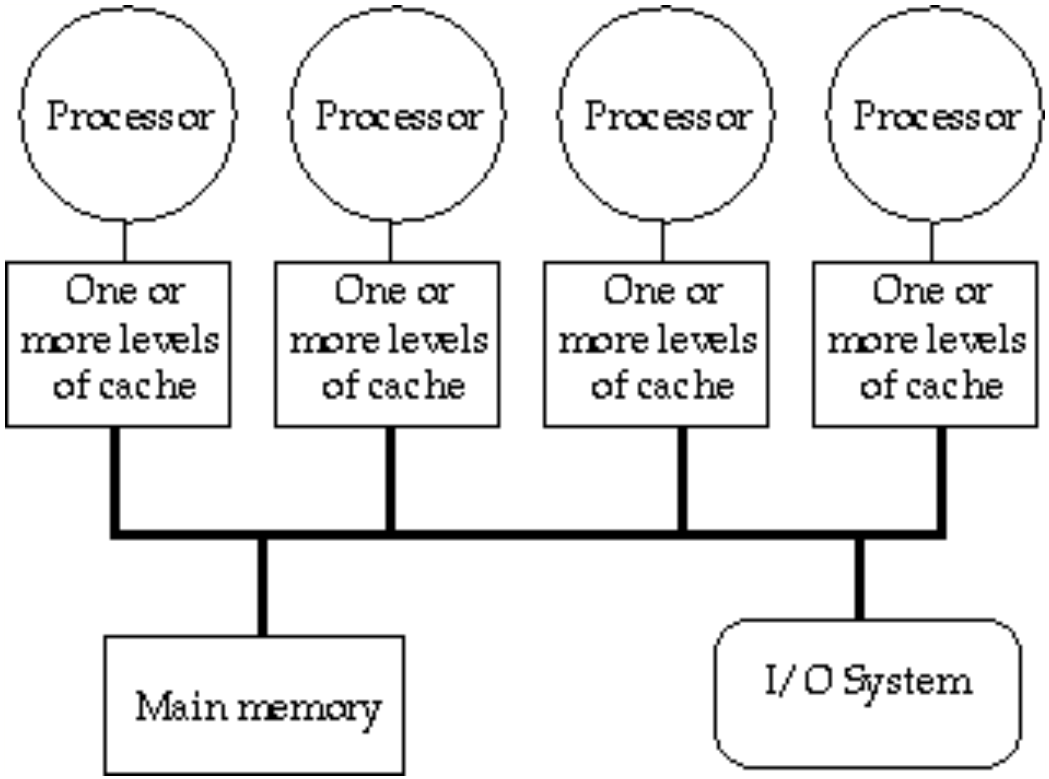
- Processadores só podem acessar memória local, e toda a comunicação e sincronização ocorrem via troca de mensagens explícita

- Fácil de se construir (máquinas uniprocessadoras + redes de comunicação)

- Escalabilidade é alta

Small-Scale—Shared Memory

- Caches servem para:
  - Reduzir necessidade de bandwidth alto entre processador/memória
  - Reduzir latência de acesso
  - Bom para dados privados e compartilhados



- Qual o problema de caches em MPs?

Coerência de Cache

Coerência de Caches

Tempo	Evento	Cache CPU A	Cache CPU B	Memória (X)
0				1
1	A Lê (X)	1		1
2	B Lê (X)	1	1	1
3	A Escreve 0 em (X)	0	1	1

Coerência de Caches

- Sistema de memória é coerente se o dado retornado após uma leitura é o dado que foi escrito mais recentemente
  - *Coerência:* Quais valores podem ser retornados durante uma leitura
  - *Consistência:* Quando valor escrito vai ser retornado durante uma escrita

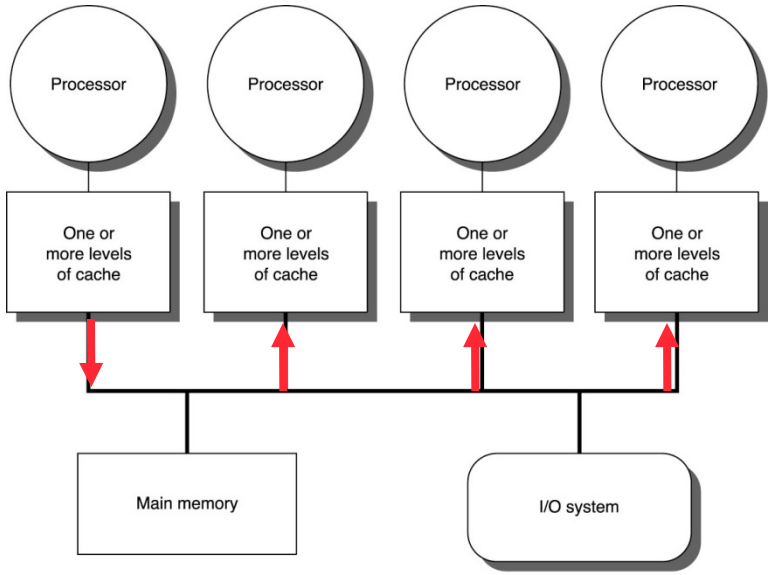
Mecanismos para Garantir Coerência de Caches

- *Snooping*: Cada cache que possui cópia própria do dado compartilhado possui também o estado do bloco, e nenhum estado centralizado é mantido.
  - Os **controladores da cache ficam** "espionando" o **barramento** compartilhado para verificar o estado atual do bloco

- *Directory based:* O estado do bloco é mantido em um diretório em uma única localização

Snoopy Caches

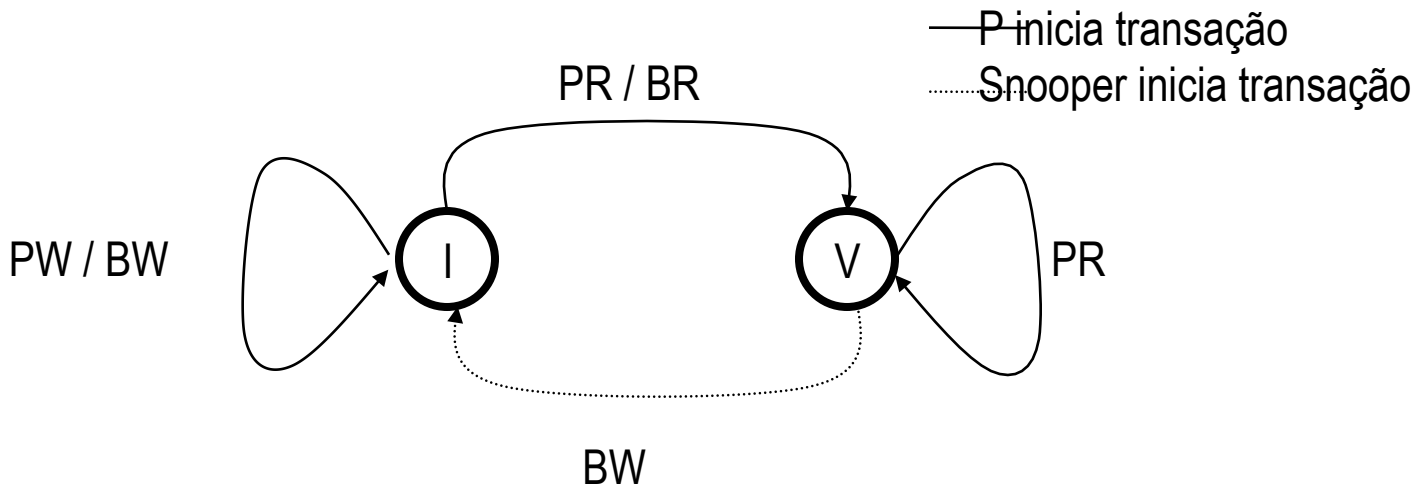
- Cada controlador "espiona" todas as transações do barramento
  - Ação depende do estado atual e do protocolo usado
- Algoritmo distribuído
  - Meio de broadcast facilita projeto, mas limita escalabilidade



Protocolos para Snoopy

- **Controlador responde** a eventos do processador e do barramento
  - **Máquina de estados finitos**
- **Opções de projeto**
  - Write-through vs. Write back
  - Invalidate vs. update

Write-through vs. Write back



- Write-through é simples, pois todo evento é observável
- Mas usa muito BW do barramento
- MPs baseados em barramento utilizam caches write-back

Coerência de Caches
Write Invalidade vs. Write Update

- *Write invalidade* (write back): Invalida todas as cópias existentes durante uma escrita
  - Trabalha a nível de bloco de cache
  - Escritas locais na cache

- *Write update* ou *write broadcast* (write through): Faz a atualização de todas as cópias durante uma escrita
  - Trabalha a nível de palavra para reduzir o tempo de escrita
  - Escritas em todas as caches
  - Reduz atraso entre escrita em uma cache e leitura em outra

Questão básica: Quando um bloco é escrito múltiplas vezes por um P antes de ser lido por outros

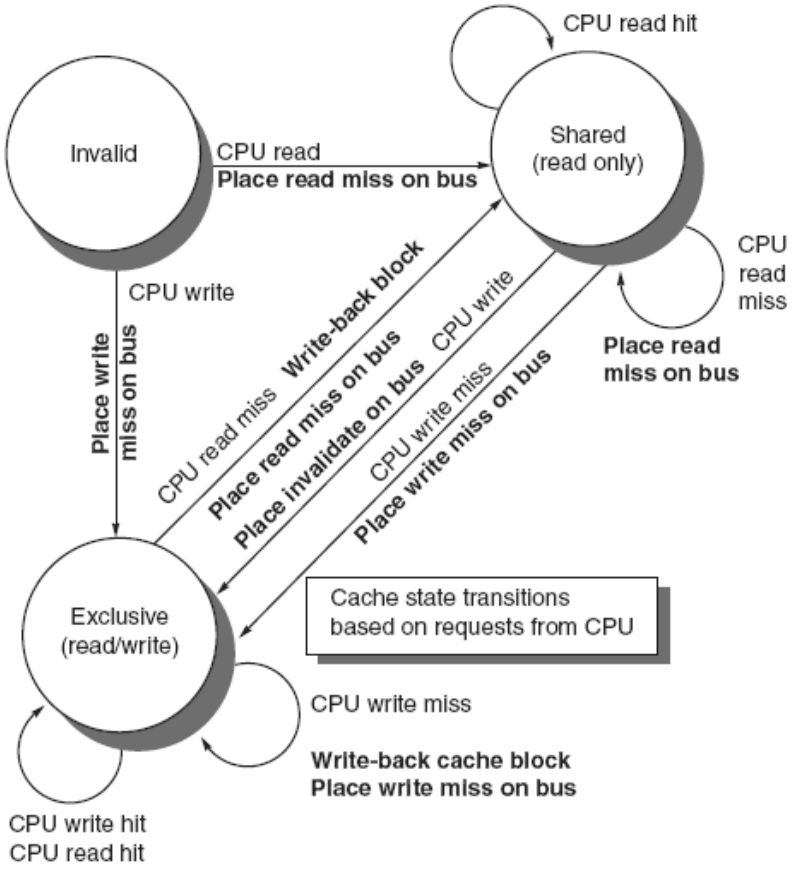
Write Invalidate

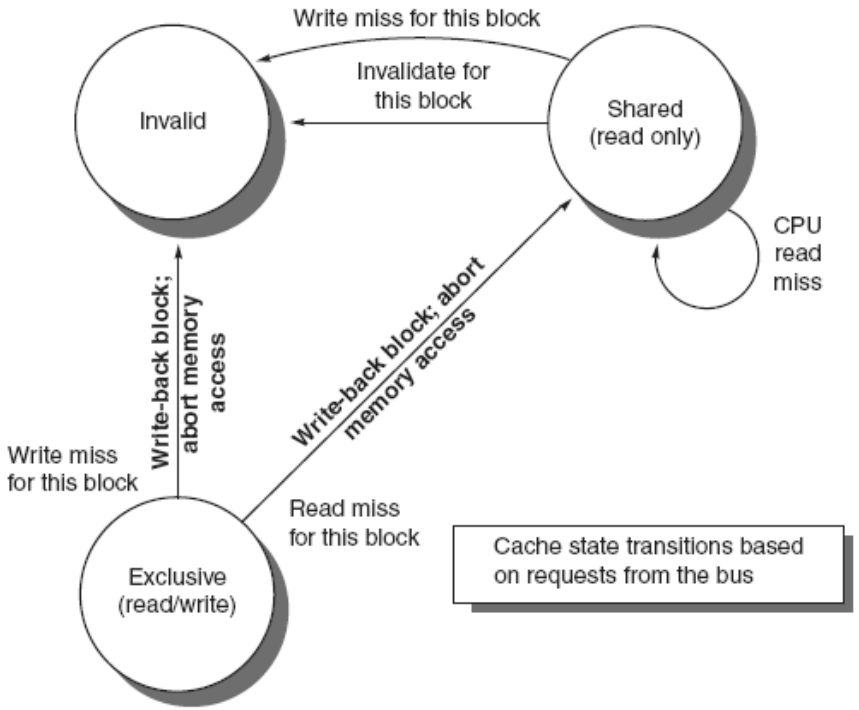
Ativ. P	Ativ. B	Cache CPU A	Cache CPU B	Memória (X)
				0
A Lê X	Miss X	0		0
B Lê X	Miss X	0	0	0
A Esc. 1	Invalid	1	I	0
B Lê X	Miss X	1	1	1

Write Update

Ativ. P	Ativ. B	Cache CPU A	Cache CPU B	Memória (X)
				0
A Lê X	Miss X	0		0
B Lê X	Miss X	0	0	0
A Esc. 1	Update	1	1	1
B Lê X		1	1	1

Protocolos de Cache Snoopy





Exemplo

Ativ. P	P1	P2	P3	Ativ. B	Fornece o Dado
P1 Lê	S	-	-	BR	Memória
P3 Lê	S	-	S	BR	Cache P1
P3 Esc.	I	-	E	BRX	
P1 Lê	S	-	S	BR	Cache P3
P2 Lê	S	S	S	BR	P1 ou P3

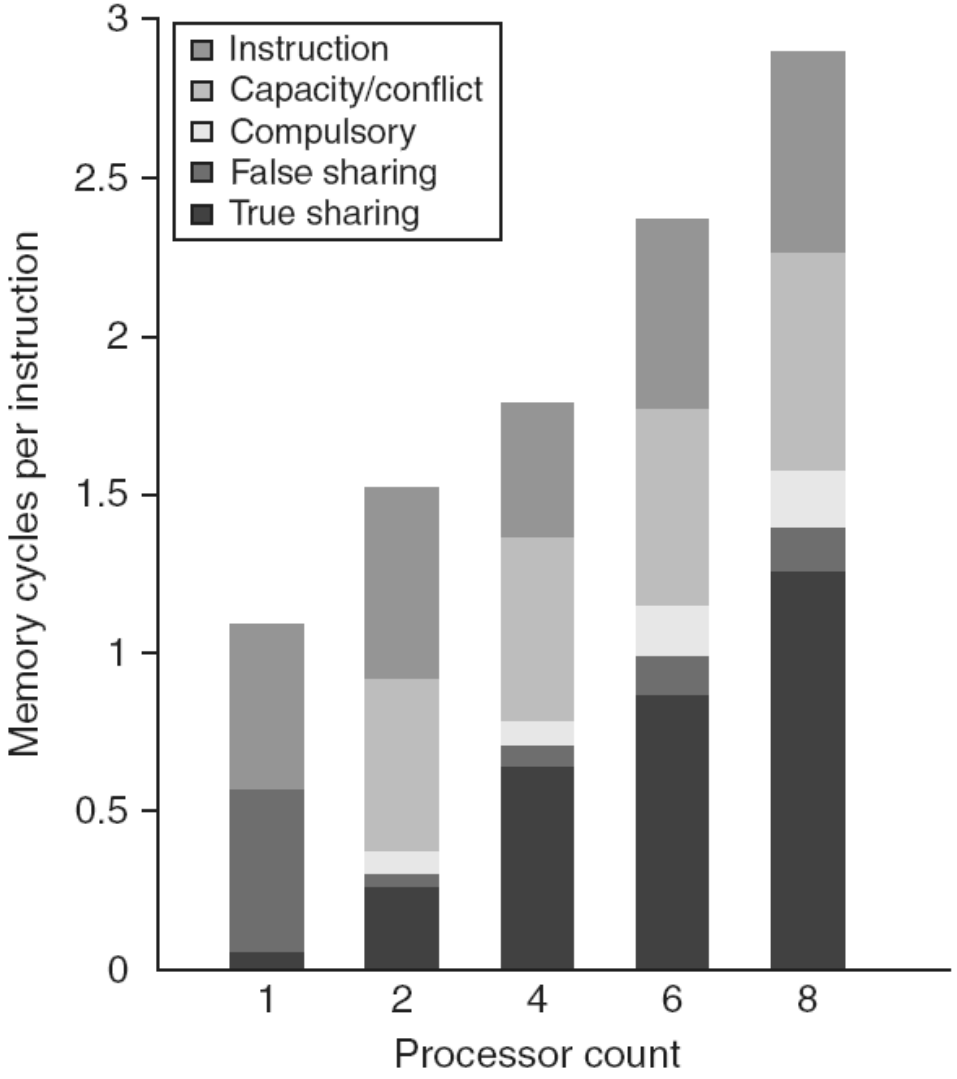
Coerência de Caches

- Classificação de MR em 3 C's
- O que pode acontecer em uma cache coerente?
  - True Sharing
    - Falha em bloco invalidado de palavra diretamente modificada.
  - False sharing
    - Falha em bloco invalidado de palavra não modificada.

Coerência de Caches

Tempo	P1	P2	Compartilhamento
1	escreve x1		
2		Lê x2	Falso
3	escreve x1		
4		escreve x2	Falso
5	Lê x2		Verdadeiro

Desempenho

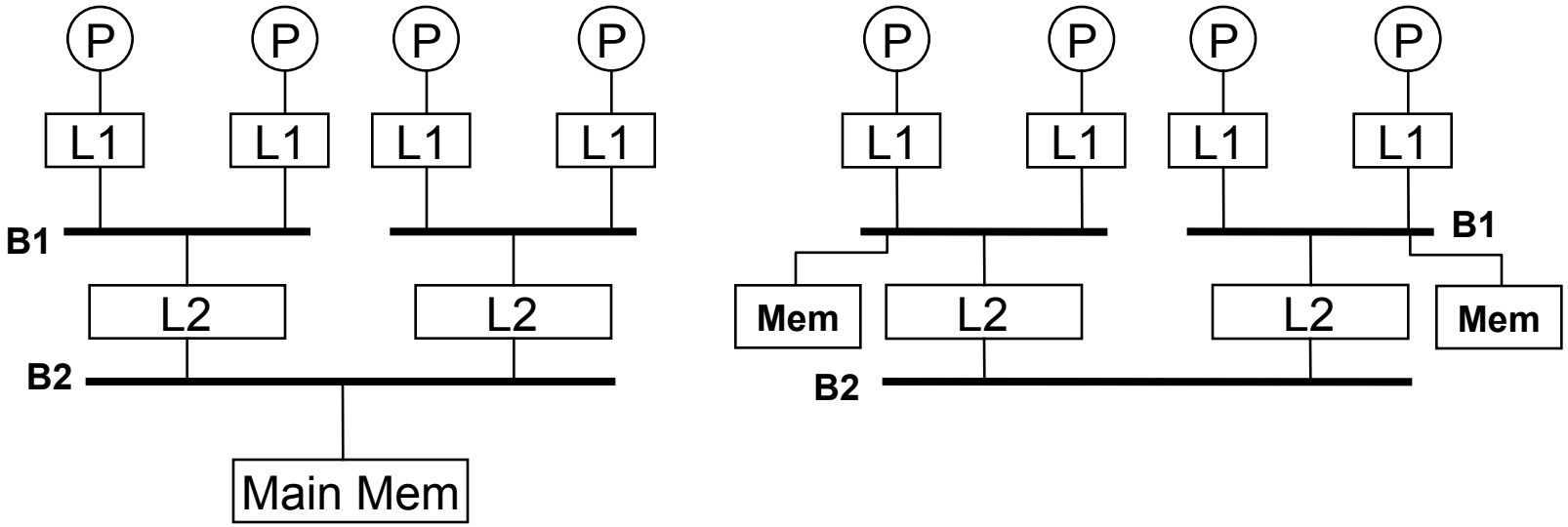

Instruction
Capacity/conflict
Compulsory
False sharing
True sharing
Memory cycles per instruction
3
2.5
2
1.5
1
0.5
0
1
2
4
6
8
Processor count

Estendendo Coerência de Caches para Memórias Compartilhadas Distribuídas

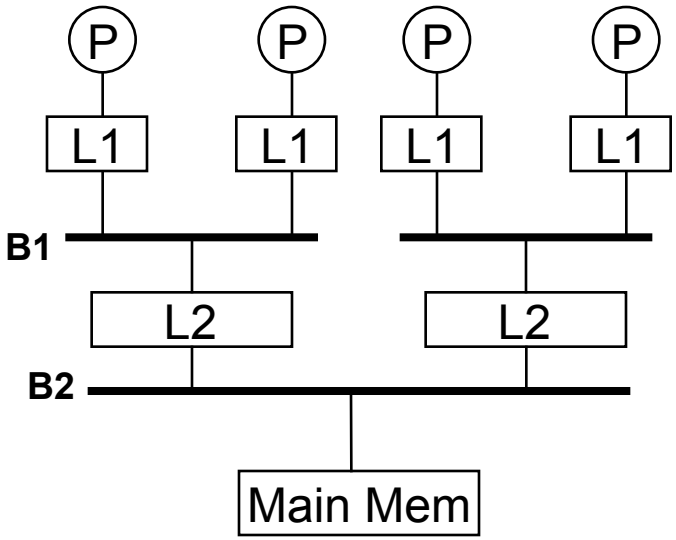
- Ao contrário de memórias compartilhadas centralizadas, não possuímos neste caso um recurso centralizado que vê todas as transações (bus)
  - Não há meio de broadcast
  - Não há maneira de serializar transações
    - barramento funciona como meio de garantir serialização das operações
- Técnicas
  - Estender protocolos do tipo snoopy
  - Protocolos baseados em diretórios

Snoopy Hierárquico

- Maneira mais simples: hierarquia de barramentos, coerência *snoopy* a cada nível
- Duas possibilidades



Hierarquias com Memória Global

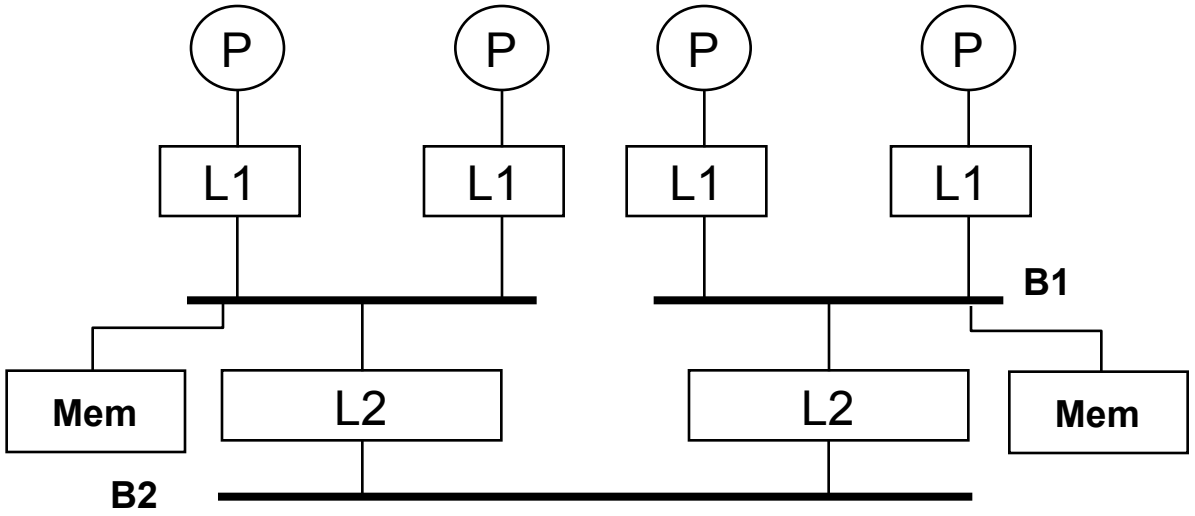


- Caches primárias:
  - Alta performance (SRAM)
  - B1 segue protocolo *snoopy* básico
- Caches secundárias:
  - Muito maiores que L1 (precisam manter propriedade de inclusão)
  - L2 funciona como filtro p/ B1 e L1
  - L2 pode ser baseada em tecnologia DRAM

Hierarquias com Memória Global

- Vantagens:
  - Misses na memória principal requerem tráfego único para a raiz da hierarquia
  - Utilização de dados compartilhados não é uma questão importante
- Desvantagens:
  - Misses para dados locais também trafegam na hierarquia
  - Memória em barramento global deve ser intercalada para aumentar BW

Hierarquias Baseadas em Clusters



- Memória principal é distribuída entre clusters
  - Alocação de dados locais pode reduzir tráfego em barramento global
  - Reduz latência (acessos locais mais rápidos)
  - Exemplos: Encore Gigamax
- L2 pode ser substituída por uma chave roteadora

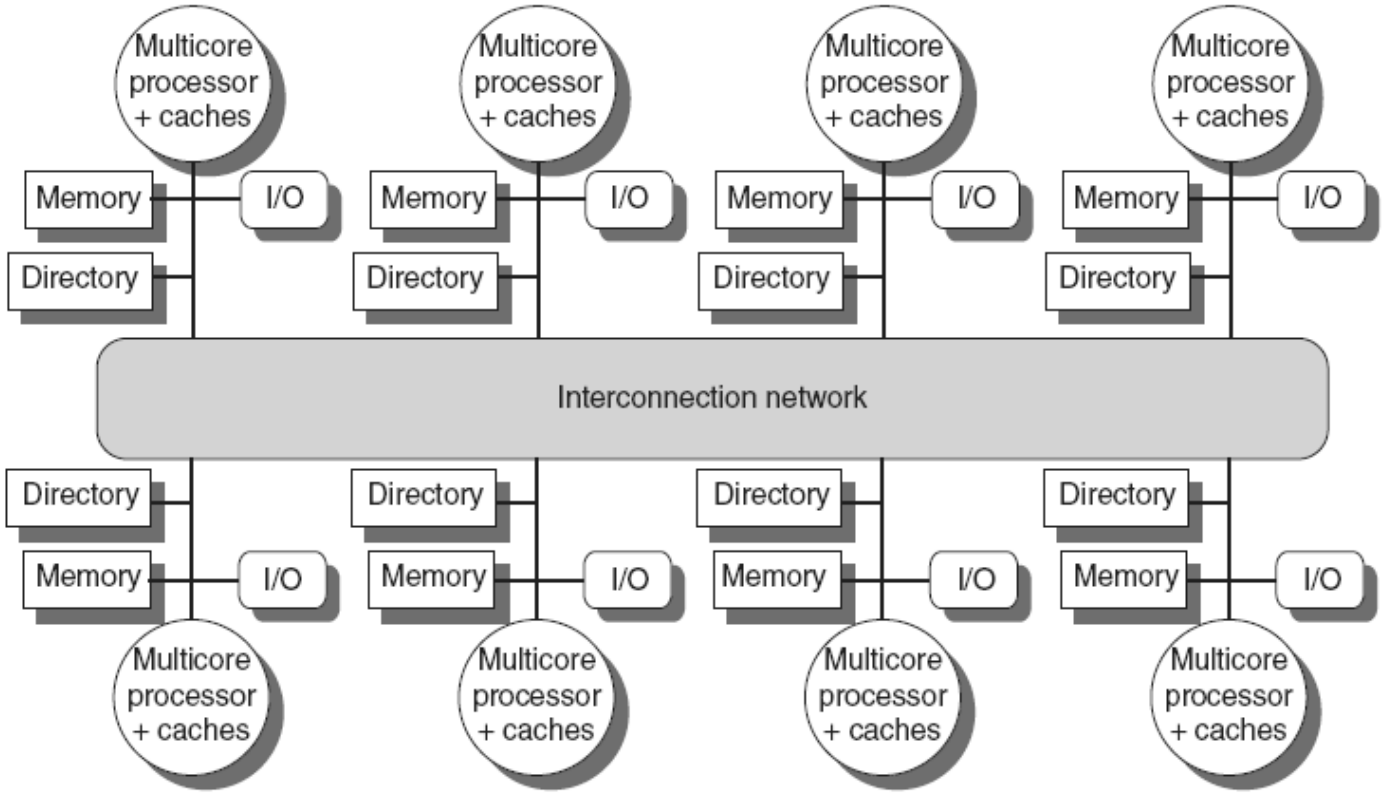
baseada em tags com coerência

Resumo de Hierarquias

- Vantagens:
  - Conceitualmente simples de se construir (aplica-se snoopy recursivamente)
  - Pode-se reduzir hardware necessário ao se combinar as funcionalidades

- Desvantagens:
  - Gargalo de comunicação na direção da raiz
  - Latências podem ser grandes na direção da raiz

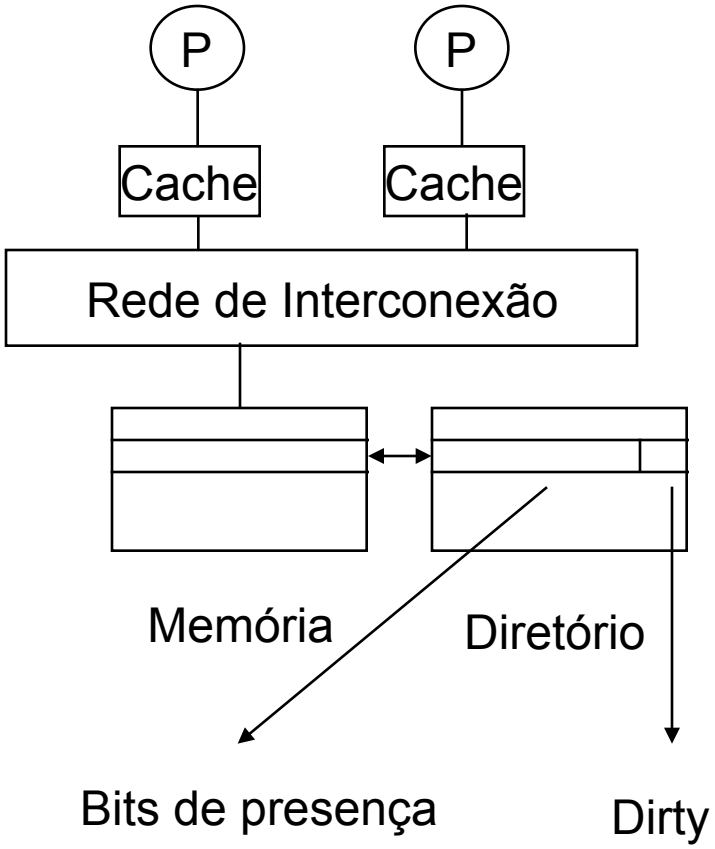
Protocolos de Coerência Baseados em Diretórios



Protocolos de Coerência Baseados em Diretórios

- Protocolos do tipo snoopy não escalam bem porque dependem de algum meio de broadcast
- Snoop hierarquico faz raiz se tornar um gargalo
- Mecanismos baseados em diretório escalam bem
  - Evitam broadcasts
    - Mantém informação de todas as caches que possuem cópia do bloco
    - Utilizam mensagens ponto-a-ponto para manter coerência

Protocolo Baseado em Diretório



- K processadores
- Para cada bloco de cache na memória => k bits de presença, 1 *dirty-bit*
- Para cada bloco na cache
  - *Invalid*
  - *Exclusive*
  - *Shared*

Estados

- *Shared:* um ou mais processadores possuem o bloco *cached*, e o bloco na memória é atualizado

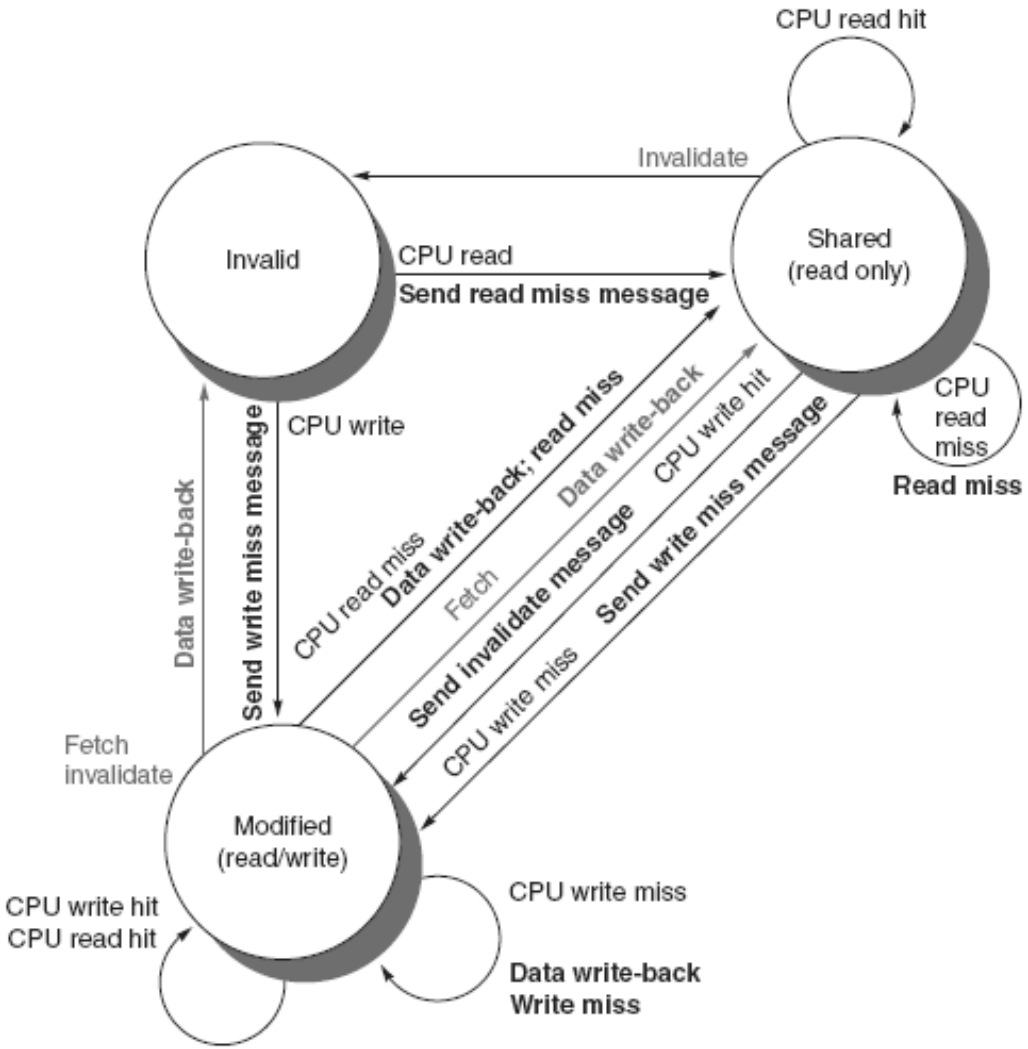
- *Invalid* ou *Uncached:* nenhum processador possui uma cópia do bloco

- *Exclusivo:* um único processador possui uma cópia do bloco

Protocolo Baseado em Diretório

Message type	Source	Destination	Message contents	Function of this message
Read miss	Local cache	Home directory	P, A	Node P has a read miss at address A; request data and make P a read sharer.
Write miss	Local cache	Home directory	P, A	Node P has a write miss at address A; request data and make P the exclusive owner.
Invalidate	Local cache	Home directory	A	Request to send invalidates to all remote caches that are caching the block at address A.
Invalidate	Home directory	Remote cache	A	Invalidate a shared copy of data at address A.
Fetch	Home directory	Remote cache	A	Fetch the block at address A and send it to its home directory; change the state of A in the remote cache to shared.
Fetch/invalidate	Home directory	Remote cache	A	Fetch the block at address A and send it to its home directory; invalidate the block in the cache.
Data value reply	Home directory	Local cache	D	Return a data value from the home memory.
Data write-back	Remote cache	Home directory	A, D	Write-back a data value for address A.

Protocolo (Cache)



Organização de Diretórios

- Implementação dos diretórios na memória (como o apresentado anteriormente)
  - Apresenta problemas de escalabilidade
  - Pode apresentar problemas de BW
- Implementação dos diretórios na cache (SCI)
  - Listas encadeadas (simples ou duplo)
  - Mantém em cada cache “link” do próximo processador compartilhando bloco
  - SCI utiliza listas duplamente encadeadas
  - Precisamos agora da mensagem “desconectar da lista” e “conectar à lista” (a última equivalente a um MISS)

Sincronização

- *Atomic exchange*
- *Test-and-set*
- *Fetch-and-increment*
- *Load-linked*

Arquiteturas Intel

Figure 4. CPU Internals of the Intel® Core™ 2 Duo Processor

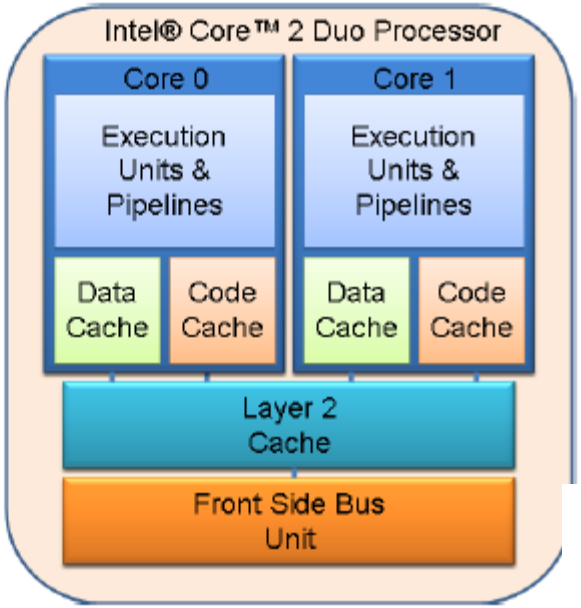


Figure 8. Intel® Core™ i7 Internals