Sistemas MIMD

CES-25 – Arquiteturas para Alto Desmpenho
Prof. Paulo André Castro
pauloac@ita.br
Sala 110 – Prédio da Computação
www.comp.ita.br/~pauloac
IEC - ITA

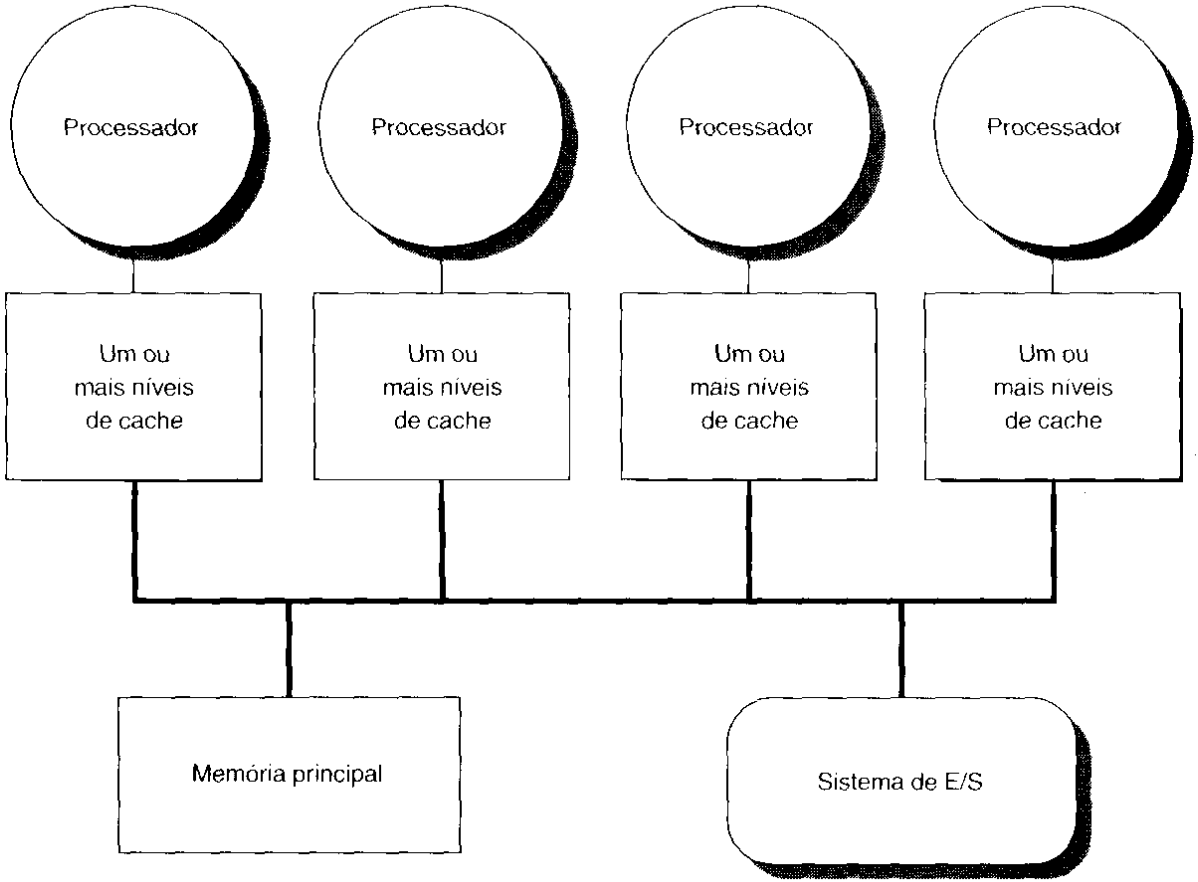
Arquiteturas Paralelas

- (SISD) Single Instruction Stream, Single Data Stream: Monoprocessador
- (SIMD) Single Instruction Stream, Multiple Data Stream: arquiteturas vetoriais
- (MISD) Multiple Instruction Stream, Single Data Stream: sem implementação comercial
- **(MIMD) Multiple Instruction Stream, Multiple Data Stream: arquiteturas multiprocessadas,**

Múltiplos Processadores

- Opção 1:
  - Compartilham cache, memória e sistema de I/O
- Opção 2:
  - Compartilham memória e sistema de I/O
- Opção 3:
  - Compartilham sistema de I/O
- Opção 4:
  - Não compartilham nada, apenas se comunicam através de redes
- Todas as opções são viáveis/interessantes?
  - Lembrem-se da importância de evitar gargalos...

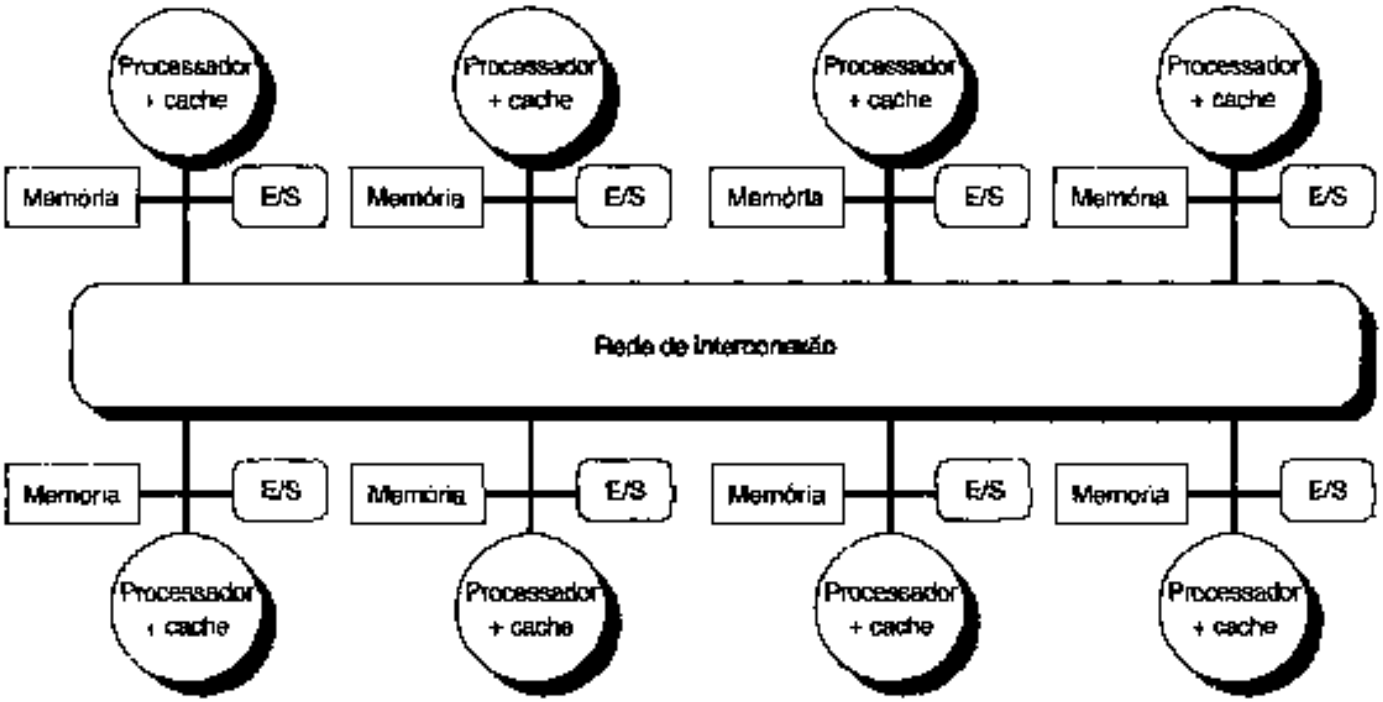
Organização Multiprocessador de Memória Compartilhada (SMP)



Memória Centralizada

- Baixo número de processadores
- A memória e seu barramento podem se tornar um gargalo para o sistema
  - uso de grandes caches e vários barramentos
- Organização mais popular atualmente

Organização Multiprocessador com Memória Distribuída



Organização Multiprocessador com Memória Distribuída

- Tipicamente maior número de processadores
- Distribuição de memória traz vantagens
  - Maior largura de banda percebidas (desde que acessos sejam principalmente locais
  - Menor latência
- Tipicamente também se distribui o sistema E/S, assim cada nó pode ser um pequeno sistema distribuído de memória centralizada

Arquitetura de Memória e Modelos de Comunicação

- Multiprocessadores simétricos (SMP) ou Uniform Memory Access(UMA) – Os processadores compartilham uma memória única e tem tempos de acesso uniforme

- Memória compartilhada Distribuída(DSM) ou Non-Uniform Access Memory (NUMA)– Os processadores compartilham o mesmo espaço de endereços, não necessariamente a mesma memória física

- Multicomputadores – processadores com memórias e espaço de endereços independentes que se comunicam através de algum tipo de rede de interconexão
  - Podem ser computadores completos ligados em rede (**clusters**)

Mecanismos de Comunicação Mensagens X Memória Compartilhada

- Passagem de Mensagens
  - Hardware mais simples
  - Comunicação Explícita, o programador controla quando ocorre ao contrário do DSM
  - Sincronização associada ao envio de mensagens
  - Facilita a comunicação iniciada pelo transmissor o que pode trazer vantagens em desempenho

Mecanismos de Comunicação 2

- Memória Compartilhada Distribuída (DSM)
  - Facilidade de programação
  - Overhead de comunicação mais baixo para itens pequenos, pela implementação em hardware e não através de E/S
  - Uso de cache pode reduzir a latência e liberar largura de banda para os demais processadores, mas introduz problemas de sincronização

Adoção no Mercado

- SMP: maior dimensão de mercado (cifras e unidades)
  - multiprocessadores em chip
- DSM (>8 processadores)
- Multicomputadores (mensagens)
  - popularização de clusters para sistemas na Internet
  - MPP (Massively Parallel Processors) > 100 processadores
    - Abordagens híbridas: mensagens e DSM

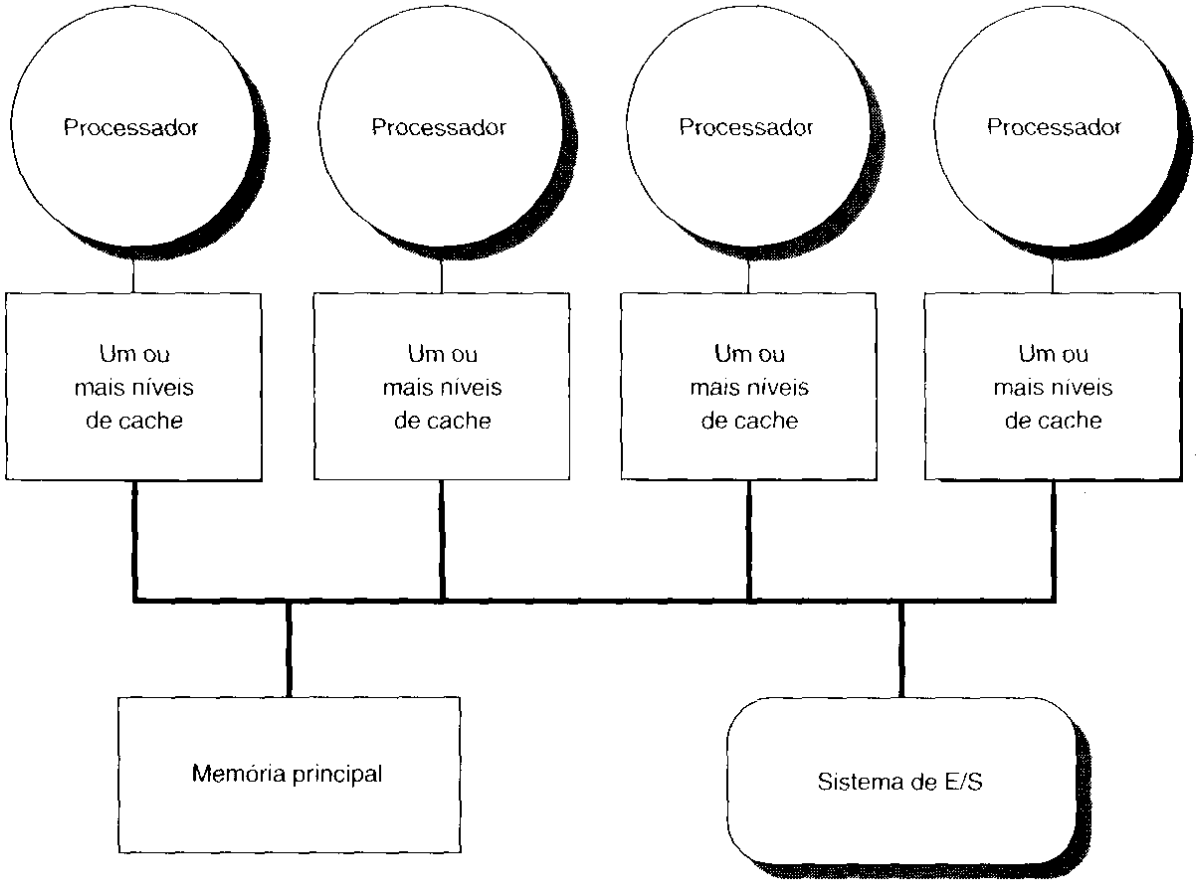
Arquitetura de Memória Compartilhada Simétrica (SMP)

- Grandes e eficientes sistemas de cache podem reduzir bastante a necessidade de largura de banda da memória

- Multiprocessadores simétricos são bastante econômicos a medida que necessitam de pouco hardware adicional e usam processadores comuns

- Em SMP, caches não apenas fornecem localidade como também replicação....Isto não traz problemas???

Organização Multiprocessador de Memória Compartilhada



Problemas em SMP ?

- Se P1 altera a posição de memória X (na sua cache) e P2 lê a posição de memória X o que ocorre?
- Isto é coerente?
- O que é coerência de caches ?
- Um sistema é coerente se ele retorna o último valor gravado em um item de dados
- Coerência e Consistência do Sistema de Memória
  - Coerência: garantir a utilização do dado mais atual
  - Consistência: sincronizar a leitura/gravação entre processadores

- Consistência :
  - P1:
    - A=0;
    - ....
    - A=1;
    - if(B==0) ...

  P2

    B=0;

    ......

    B=1;

    if(A==0) ....
- Inicialmente, A e B em cache com valor igual a zero, qual dos dois if é seguido ou os dois?
- Muitas vezes precisa ser tratado pelo próprio programador

Coerência

- Um sistema de memória é coerente se:
  1. Uma leitura por um processador P em uma posição X seguido de um gravação por P em X, sem a ocorrência de gravações em X por outro processador neste intervalo, sempre retorna o valor gravado por P.
  2. Uma leitura por P1 na posição X após uma gravação por P2 em X retorna o valor gravado se a leitura e a gravação estiverem separadas no tempo e não ocorrer nenhuma outra gravação em X entre os dois acessos.
  3. Gravações na mesma posição são serializadas; isto é, duas gravações na mesma posição por dois processadores quaisquer são vistas na mesma ordem por todos os processadores. Por exemplo, se os valores 1 e depois 2 são gravados em X, nenhum processador pode ler 2 e depois 1

Caches Coerentes

- Manutenção em Hardware da coerência de caches através de **protocolos de coerência de Cache**

- Abordagem baseada em Snooping
  - Invalidação de gravação
  - Atualização ou Difusão

- Abordagem baseada em Diretório
  - Usadas em Arquitetura de memória distribuída compartilhada

Protocolos de Snooping

- Cada cache tem um cópia dos dados de um bloco de memória e também uma cópia do status de compartilhamento do bloco(compartilhado/ não compartilhado)

- Como as caches compartilham o barramento de memória elas espionam (*snoop*) o tráfego para verificar se tem cópias do bloco trafegado
- Snooping com Invalidação
  - Gravação em bloco compartilhado invalida as demais cópias do bloco em cache.
  - Ao tentar acessar um bloco inválido, há uma falha de cache, e o dado vem do bloco de cache “dirty” e também para a memória (caso write-back)
  - Gravações em blocos não compartilhados não geram problemas. Porque?
  - O que aconteceria com sistemas write-through?

Protocolos de Snooping com Invalidação

Atividade do processador	Atividade do barramento	Conteúdo da cache da CPU A	Conteúdo da cache da CPU B	Conteúdo da Posição de Memória X
				0
CPU A lê X	Erro de cache para X	0		0
A CPU B lê X	Erro de cache para X	0	0	0
A CPU A grava 1 em X	Invalidação de X	1		0
A CPU B lê X	Erro de cache para X	1	1	1

Protocolos de Snooping

- Snooping com Atualização ou Difusão
  - Diferença apenas no tratamento da gravação, o armazenamento de cache é o mesmo. Isto é, bloco e status do bloco
  - Gravação em bloco compartilhado atualiza as demais cópias do bloco em cache e também a memória
- Exemplo

Atividade do processador	Atividade do barramento	Conteúdo da cache da CPU A	Conteúdo da cache da CPU B	Conteúdo da posição de memória X
				0
CPU A lê X	Falha de cache para X	0		0
A CPU B lê X	Falha de cache para X	0	0	0
A CPU A grava 1 em X	Difusão de gravação de X	1	1	1
A CPU B lê X		1	1	1

Comparativo
Invalidação X Atualização

Pró-Atualização

1. O retardo entre a gravação de uma palavra em um processador e a leitura do valor gravado em outro processador é **menor** em um esquema de atualização
   - Pois na atualização, os dados gravados são atualizados imediatamente na cache do leitor.

Comparativo
Invalidação X Atualização

Pró-Invalidação

1. Várias gravações da mesma palavra sem leituras intervenientes exigem várias difusões de gravação, mas apenas uma invalidação inicial
2. Cada palavra gravada em um bloco de cache exige uma difusão de gravação em um protocolo de atualização, embora apenas a primeira gravação de qualquer palavra no bloco precise gerar uma invalidação.
   - Um protocolo de invalidação atua sobre blocos de cache, enquanto um protocolo de atualização deve atuar sobre palavras individuais
3. Protocolos de invalidação tendem a gerar menos tráfego no barramento de memória (item que tende a ser um gargalo em SMP)

Invalidação

- Por tais razões, o protocolo de Snooping por Invalidação tornou-se o mais popular, quase todos os SMP usam invalidação

Implementação da Invalidação

- Invalidação dos blocos
  - A chave da implementação é obter acesso ao barramento e utilizá-lo para invalidar um bloco, o processador envia o endereço do bloco pelo barramento
  - Os demais processadores estão espiando (snooping) o barramento e verificam se tem aquele bloco em suas caches, checando o endereço e invalidando o bloco
- Gravação serializada
  - A necessidade de obter acesso ao barramento (recurso exclusivo) força a serialização das gravações

Estados de um Bloco – Snooping por Invalidação

- O Estado de um bloco é único para todo o conjunto de caches. Os estados possíveis são:
- **Inválido**: não está em cache
- **Exclusivo**: Está apenas em uma cache (não necessariamente na cache do processador que solicitou)
- **Compartilhado**: Está em uma ou mais de uma cache (não necessariamente na cache do processador que solicitou)

- O estado passa de **Inválido** para **compartilhado** já na primeira leitura do bloco (mesmo que exista apenas uma cópia), na primeira gravação passa para exclusivo.

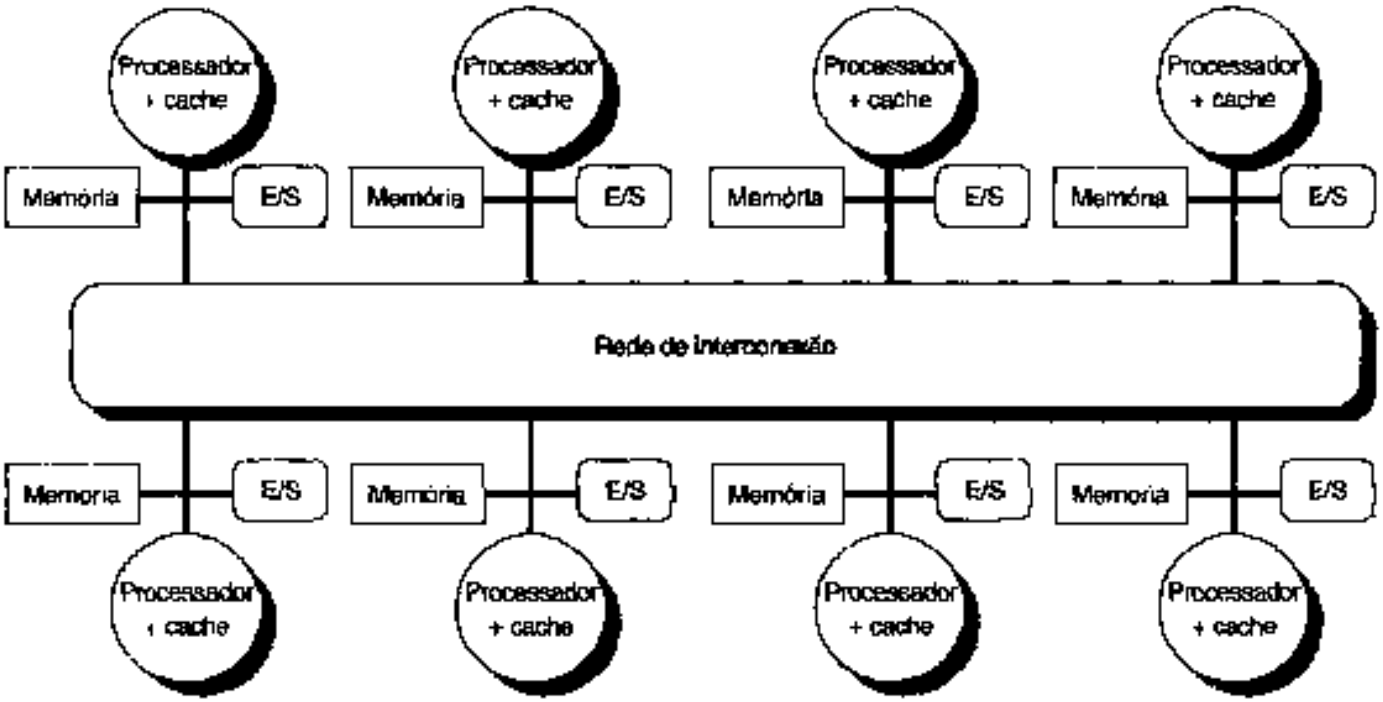
Implementação da Invalidação

- Localização de dados em uma falha de cache
  - Write Through: Busca o dado da memória, não é necessário gravar blocos de cache na memória
- Write Back: ?
  - Se não sujo (compartilhado): funciona como write through
  - Se sujo (exclusivo): Cancela a operação de memória e envia para o processador requisitante seu bloco em geral também envia para a memória neste momento.

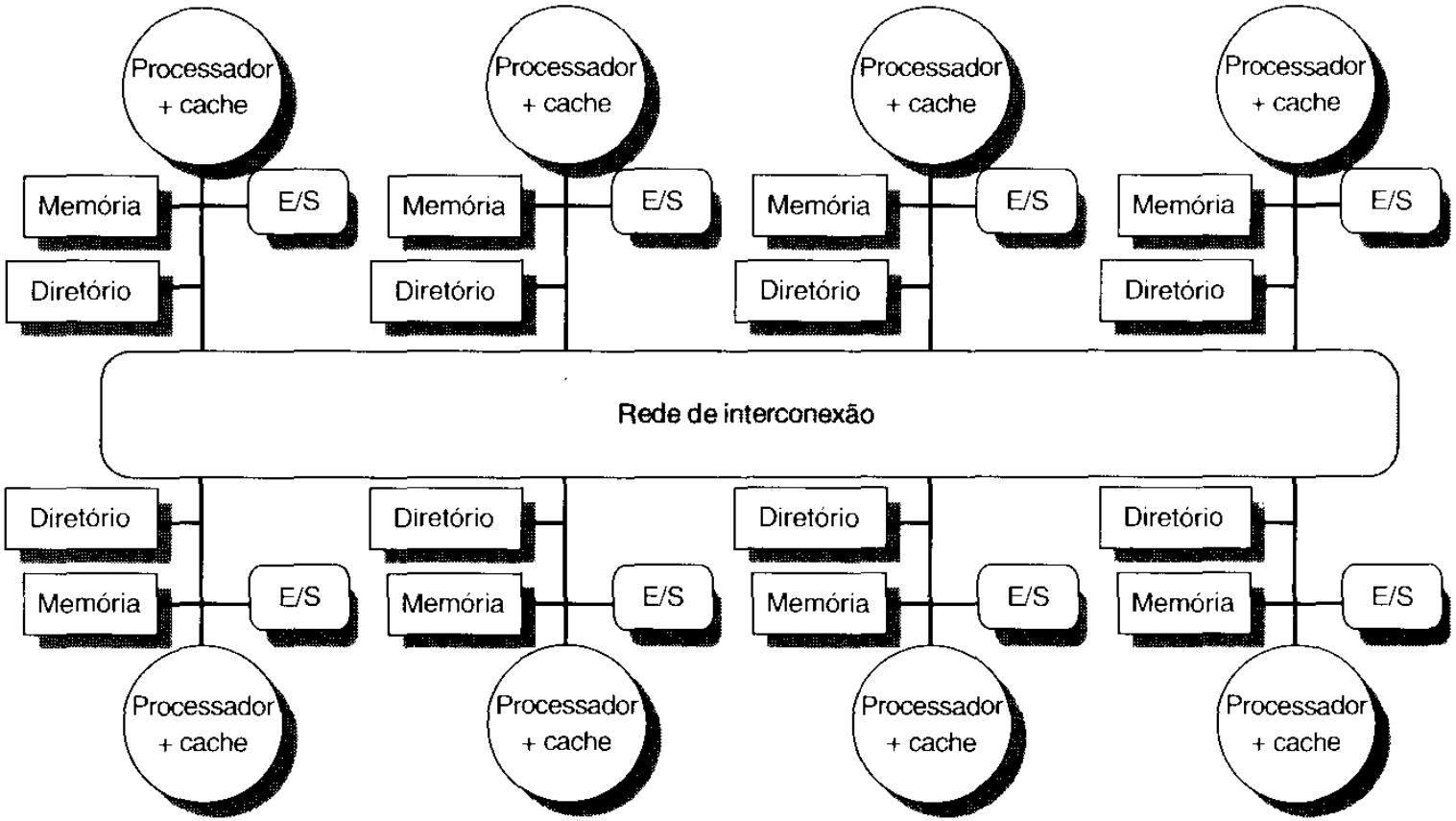
Protocolo de Invalidação

Solicitação	Origem	Estado do bloco de cache endereçado	Função e explicação
Acerto de leitura	processador	compartilhado ou exclusivo	Ler dados em cache.
Falha de leitura	processador	inválido	Colocar falha de leitura em barramento.
Falha de leitura	processador	compartilhado	Erro de conflito de endereço: colocar falha de leitura em barramento.
Falha de leitura	processador	exclusivo	Erro de conflito de endereço: gravar bloco de volta, depois colocar erro de leitura em barramento.
Acerto de gravação	processador	exclusivo	Gravar dados em cache.
Acerto de gravação	processador	compartilhado	Colocar erro de gravação em barramento.
Falha de gravação	processador	inválido	Colocar erro de gravação em barramento.
Falha de gravação	processador	compartilhado	Erro de conflito de endereço: colocar erro de gravação em barramento.
Falha de gravação	processador	exclusivo	Erro de conflito de endereço: gravar bloco de volta, depois colocar erro de gravação em barramento.
Falha de leitura	barramento	compartilhado	Nenhuma ação; permitir à memória atender a erro de leitura.
Falha de leitura	barramento	exclusivo	Tentar compartilhar dados: colocar bloco de cache em barramento e alterar estado para compartilhado.
Falha de gravação	barramento	compartilhado	Tentar gravar bloco compartilhado; invalidar o bloco.
Falha de gravação	barramento	exclusivo	Tentar gravar bloco que é exclusivo em outro lugar: gravar de volta o bloco da cache e tornar seu estado inválido.

Problema de Coerência de Cache em DSM com Espaço de endereços compartilhados?



DSM com Protocolo de Coerência por Diretório



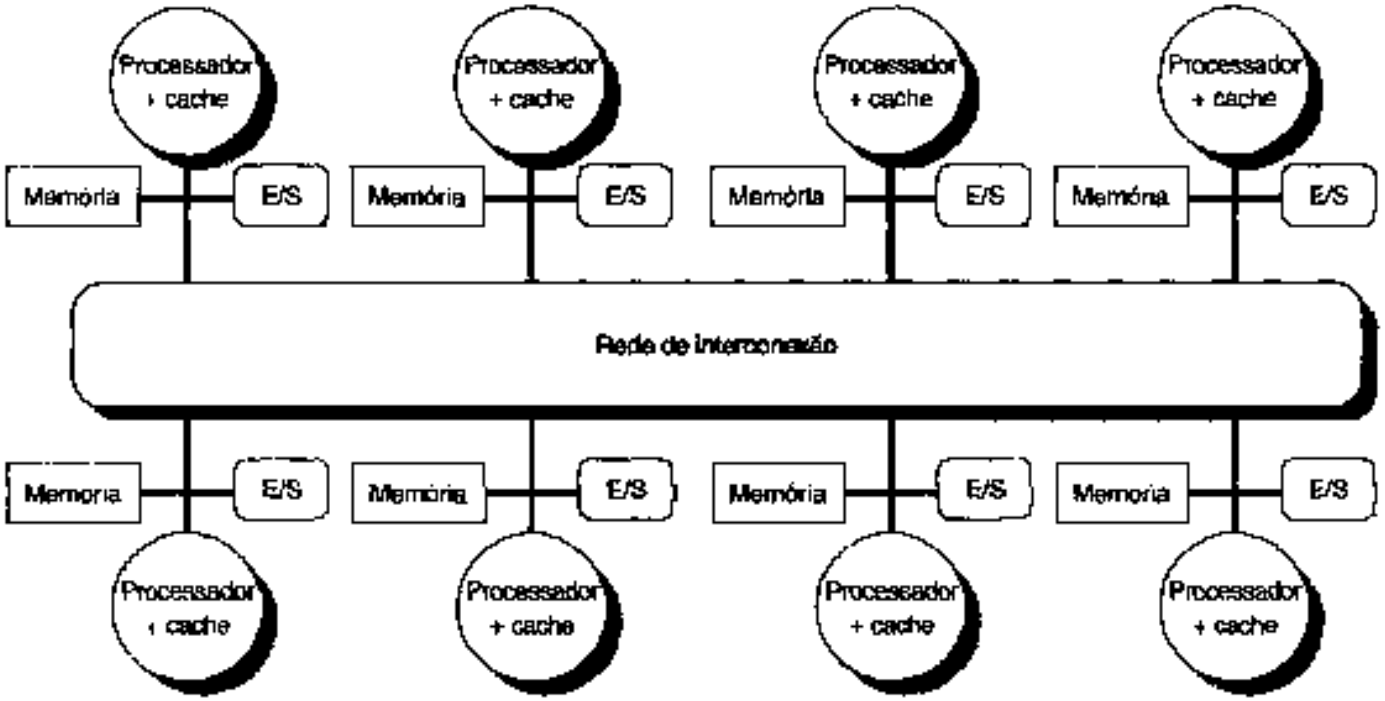
Diretórios

- Diretórios mantêm o estado de todo bloco que pode ser inserido na cache,
  - as identificação das caches que tem cópias do bloco
  - se o bloco está sujo
  - Se o bloco é compartilhado, exclusivo ou não-inserido em cache
- O diretório faz o papel dos campos de status no protocolo de snooping

Diretórios

- Os diagramas de estado são os mesmos utilizados no Snooping
- Implementação é feita com base em troca de mensagens entre os nós, ao invés de espionagem do barramento
  - Necessário saber quem tem o bloco para fazer a invalidação
- Cada processador mantêm a informação de quais processadores tem cada bloco de memória.
  - Campo com um bit associado para cada processador do sistema para cada bloco de memória

Multicomputadores e Coerência de Cache



- Há problema de coerência de caches em sistemas multiprocessados baseados em mensagens?

Paralelismo em Nível de Thread

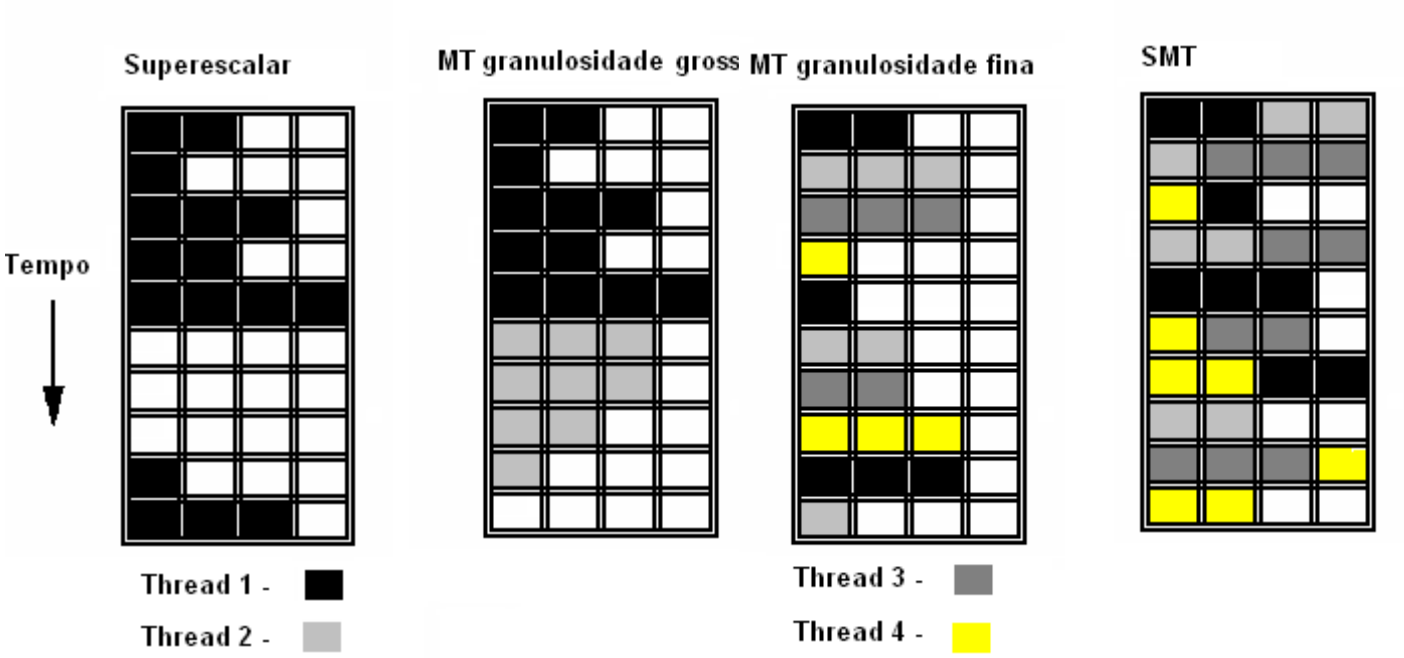
Paralelismo em Nível de Thread (TLP)

- É cada vez mais comum programas utilizarem multithreading

- Threads são uma forma de paralelismo explícito e “controlado” pelo programador
  - Diferente de ILP e instruções vetoriais
  - Porque não explorar para ganhar desempenho?

- Qual a relação entre Processadores Superescalares e Multithreading ?
  - SMT (Simultaneous Multithreading)

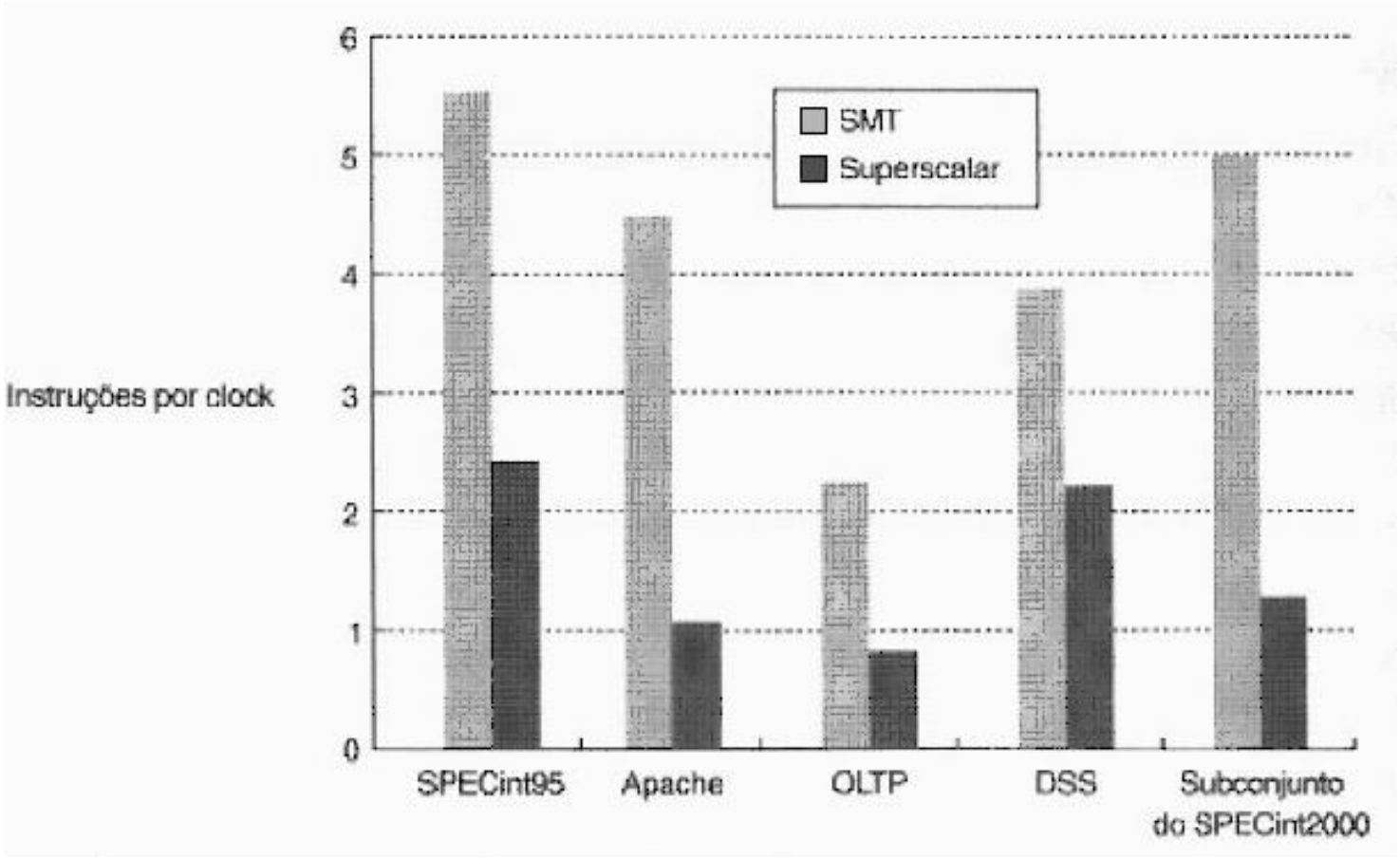
Superescalar e SMT com 4 pipelines



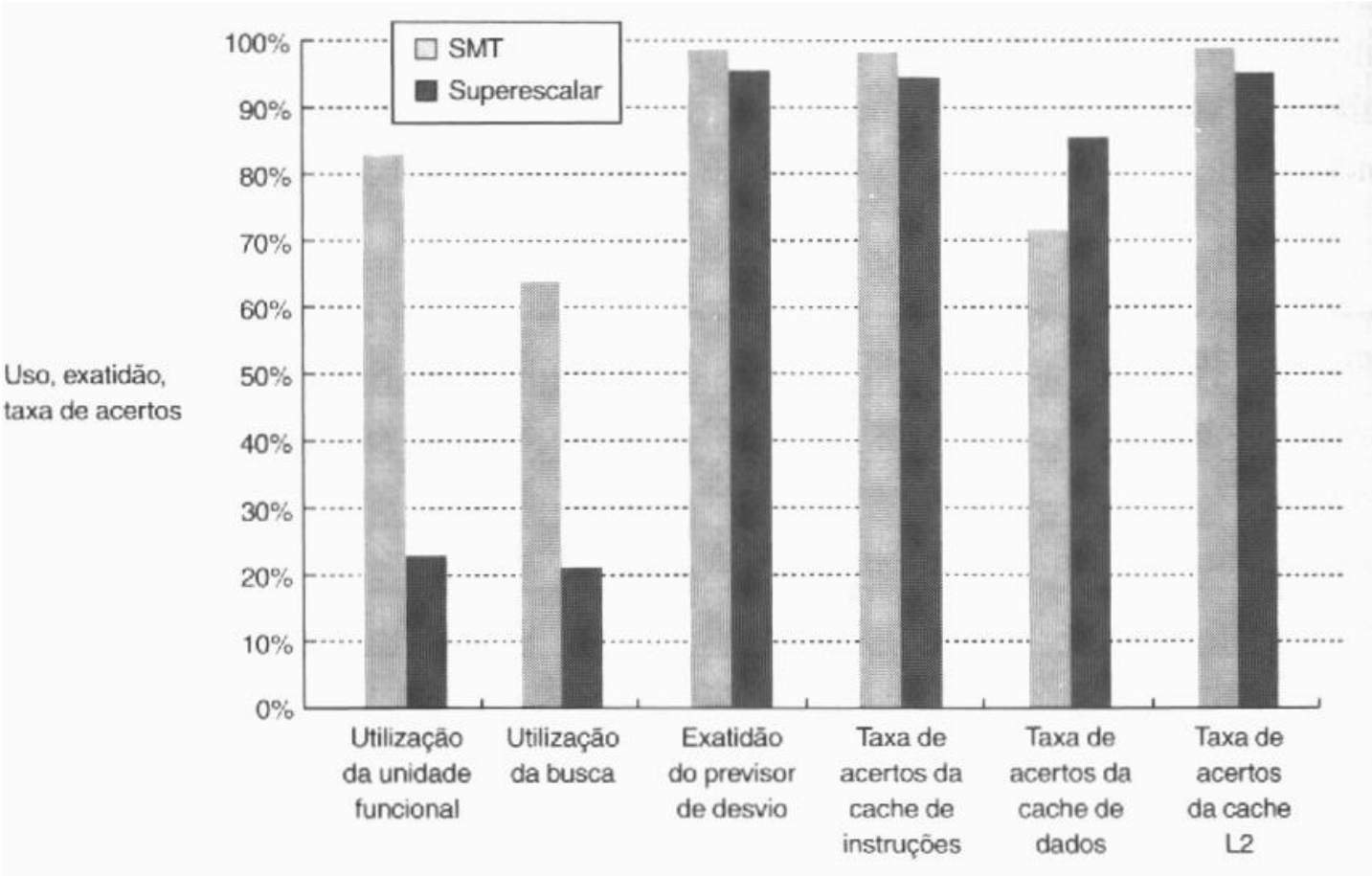
Um Processador SMT Teórico

Característica do processador	Capacidade
Unidades funcionais de inteiros	6, incluindo até 4 instruções de carga/armazenamento por ciclo
Profundidade do pipeline	9 fases
Unidades funcionais de ponto flutuante	4 (todas de uso geral)
Filas de instruções	32 entradas cada, para números inteiros e de ponto flutuante
Registradores de mudança de nomes	100 cada, para números inteiros e de ponto flutuante (além dos registradores arquitetônicos)
Capacidade de consolidação	Até 12 instruções por ciclo
TLB	128 entradas cada, para instruções e dados
Cache de instruções primária	128 KB, associativa de conjunto de duas vias, de porta única, penalidade de preenchimento de dois ciclos, 32 erros pendentes (compartilhada com a cache de dados primário)
Cache de dados primária	128 KB, associativa de conjunto de duas vias, de porta dual, penalidade de preenchimento de dois ciclos
Cache L2	16 MB, de mapeamento direto, latência de 20 ciclos, com pipeline total
Barramento/reposição L1-L2	256 bits de largura, latência de 2 ciclos
Buffer de armazenamento	32 entradas
Sistema de memória	latência de 90 ciclos, com pipeline total, conectado a um barramento de 128 bits de largura e latência de 4 ciclos
Previsor de desvio	Previsor de torneio com previsor local de 4K indexado por uma tabela de histórico com 2K entradas, um previsor global de 8K e uma tabela de seletores com 8K entradas
Buffer de destino de desvio	1K entrada, associativo de conjunto de quatro vias
Contextos de hardware para SMT	8
Norma de busca	8 instruções por clock, de até 2 contextos

Desempenho (throughput) Simulado - SMT x Superescalar



Outras Medidas
SMTx Superescalar



Resumo

- Os ganhos em Throughput são impressionantes(Hennesy, p. 451) de 1,7 a 4,2 vezes com média de 3,0 vezes

- O processador usado tem algumas características relevantes: grandes caches primárias, uma cache secundária agressiva e grandes números de unidades funcionais. Capacidade para até 8 contextos de Threading

- O SMT foi amplamente adotado em vários sistemas, mesmo em hardware mais modesto