Universidade Federal do Rio de Janeiro
Pós-Graduação em Informática
DCC/IM - NCE/UFRJ

Arquiteturas de Sistemas de Processamento Paralelo

Protocolos de Coerência de Memória Cache

Gabriel P. Silva

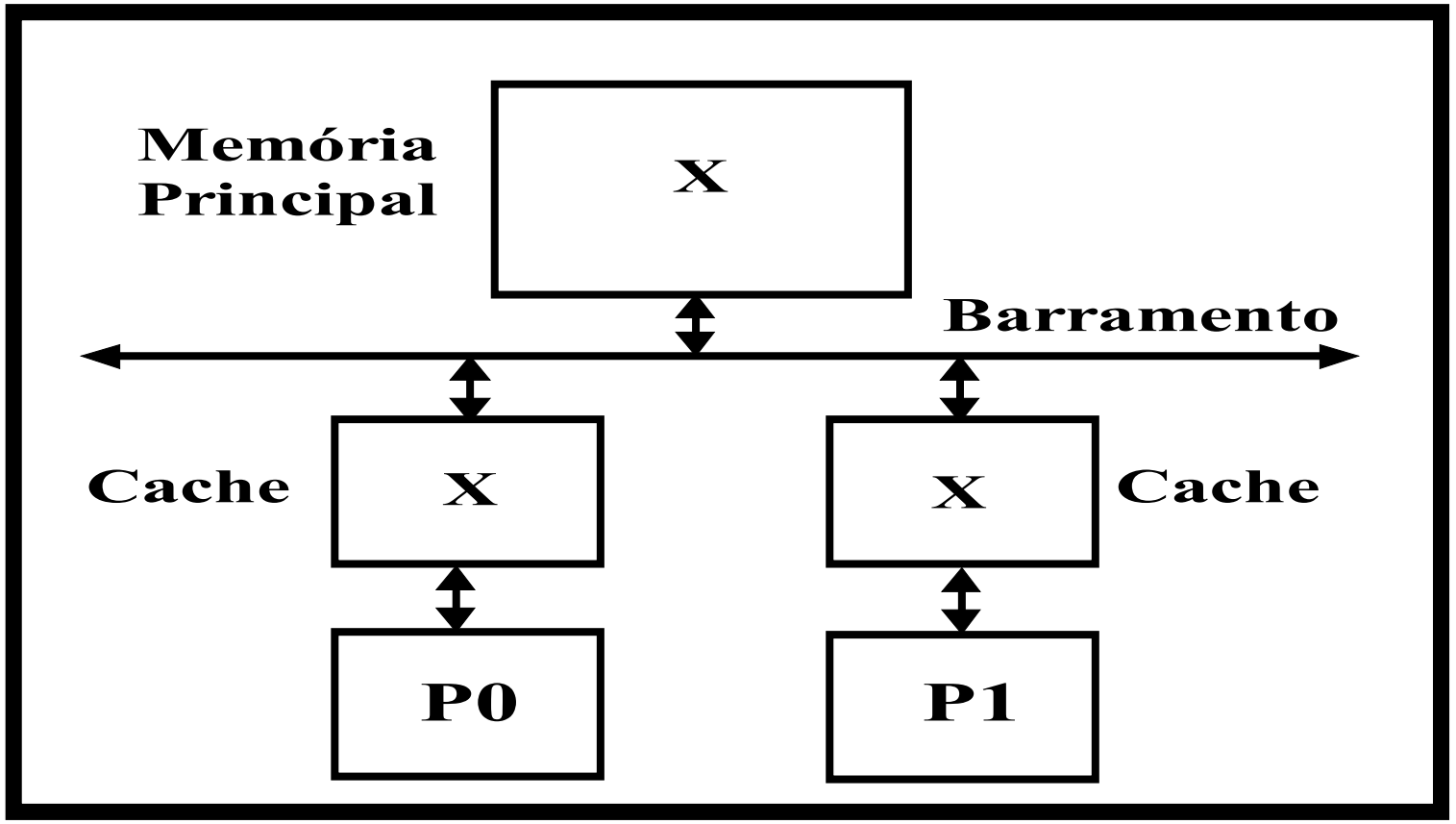
Introdução

- Em sistemas com memória compartilhada em que os processadores fazem uso de caches é necessário manter a consistência dos dados armazenados nessas caches.
- Existem dois tipos básicos de protocolo de coerência de cache:
  - Para sistemas que fazem uso de barramento compartilhado.
  - Para sistemas que fazem uso de redes de interconexão.
- O principal problema ocorre quando um processador modifica (escreve) em um dado que está em um bloco (linha) armazenado na sua cache.

Introdução

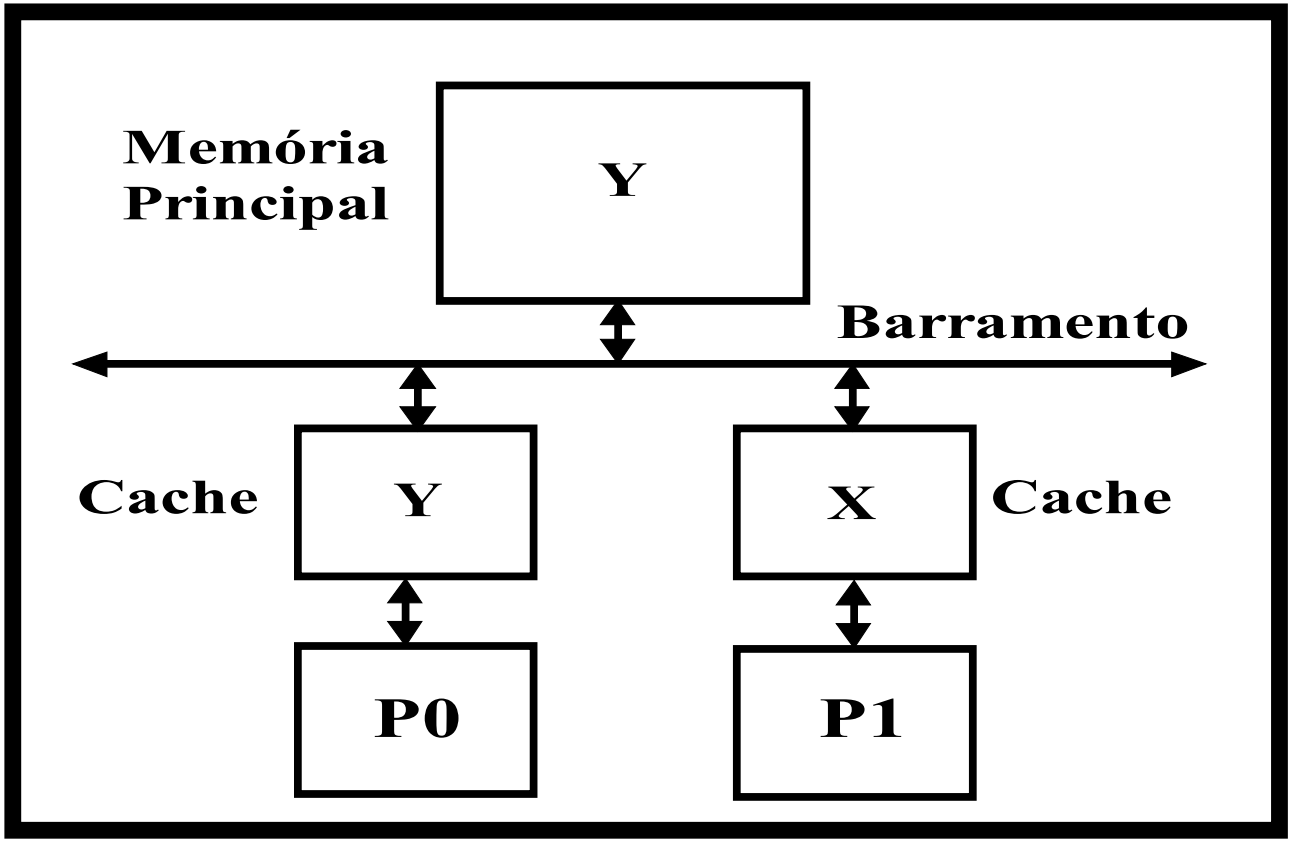
- As memórias caches possuem dois modos básicos para trabalhar em relação à atualização dos dados na memória principal durante uma escrita:
  - Write-through → Os dados são atualizados tanto na memória cache como na memória principal.
  - Write-back → Os dados são atualizados apenas na memória cache, e copiados para a memória principal, apenas quando da substituição do bloco/linha modificado na cache.

Introdução



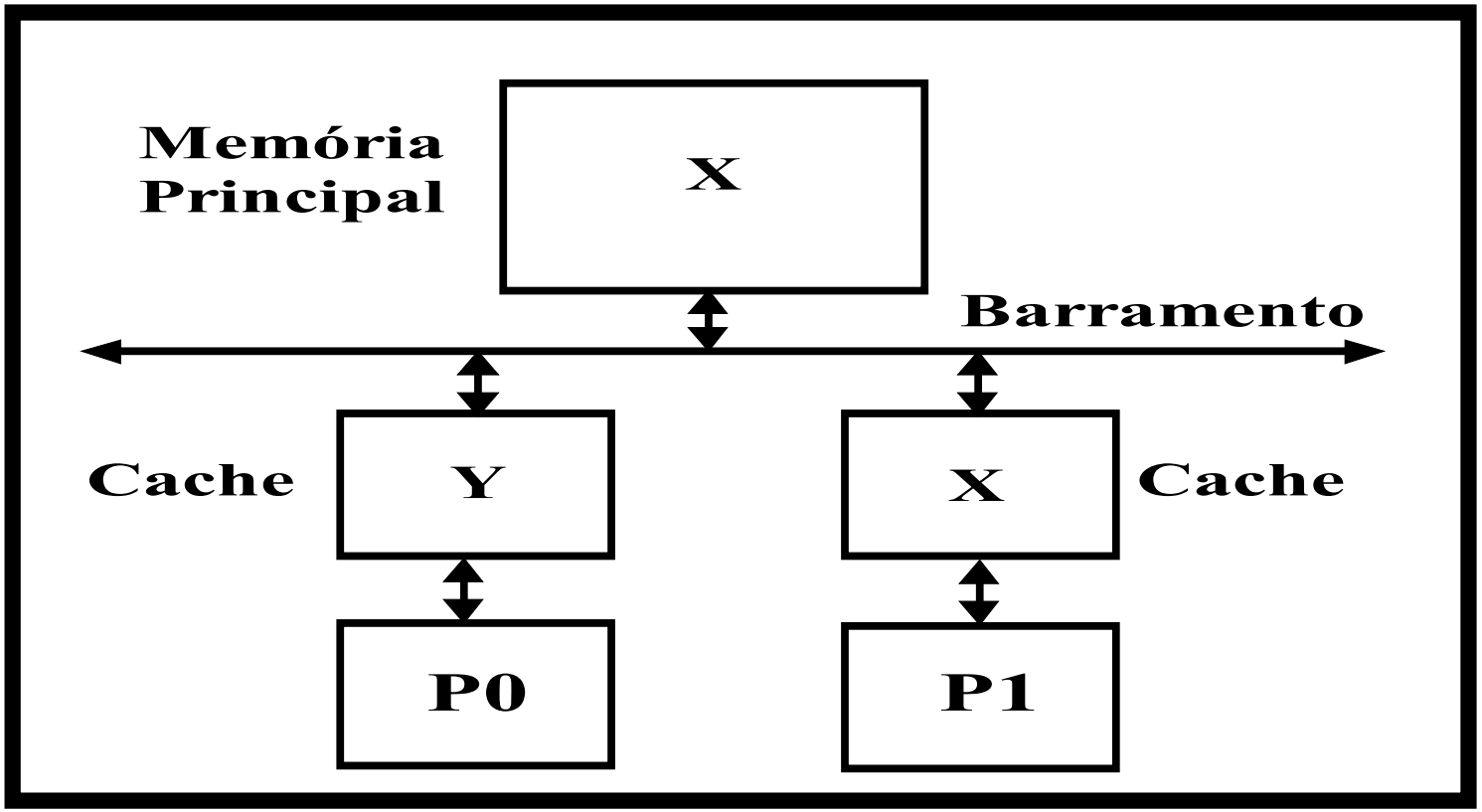
(a) Situação Inicial

Coerência de Cache



(b) Escrita no Modo "Write-Through"

Coerência de Cache

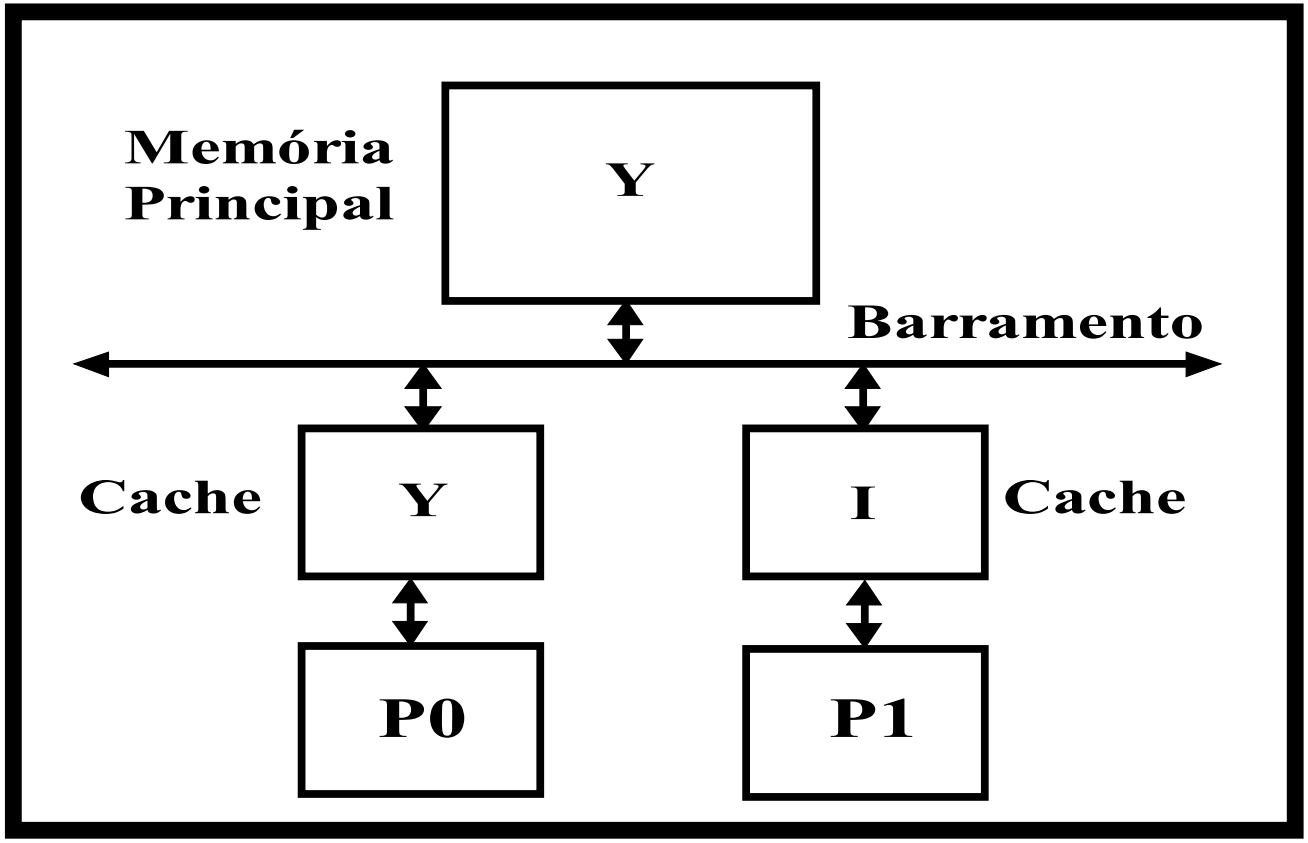


(c) Escrita no Modo "Write-Back"

Coerência de Cache

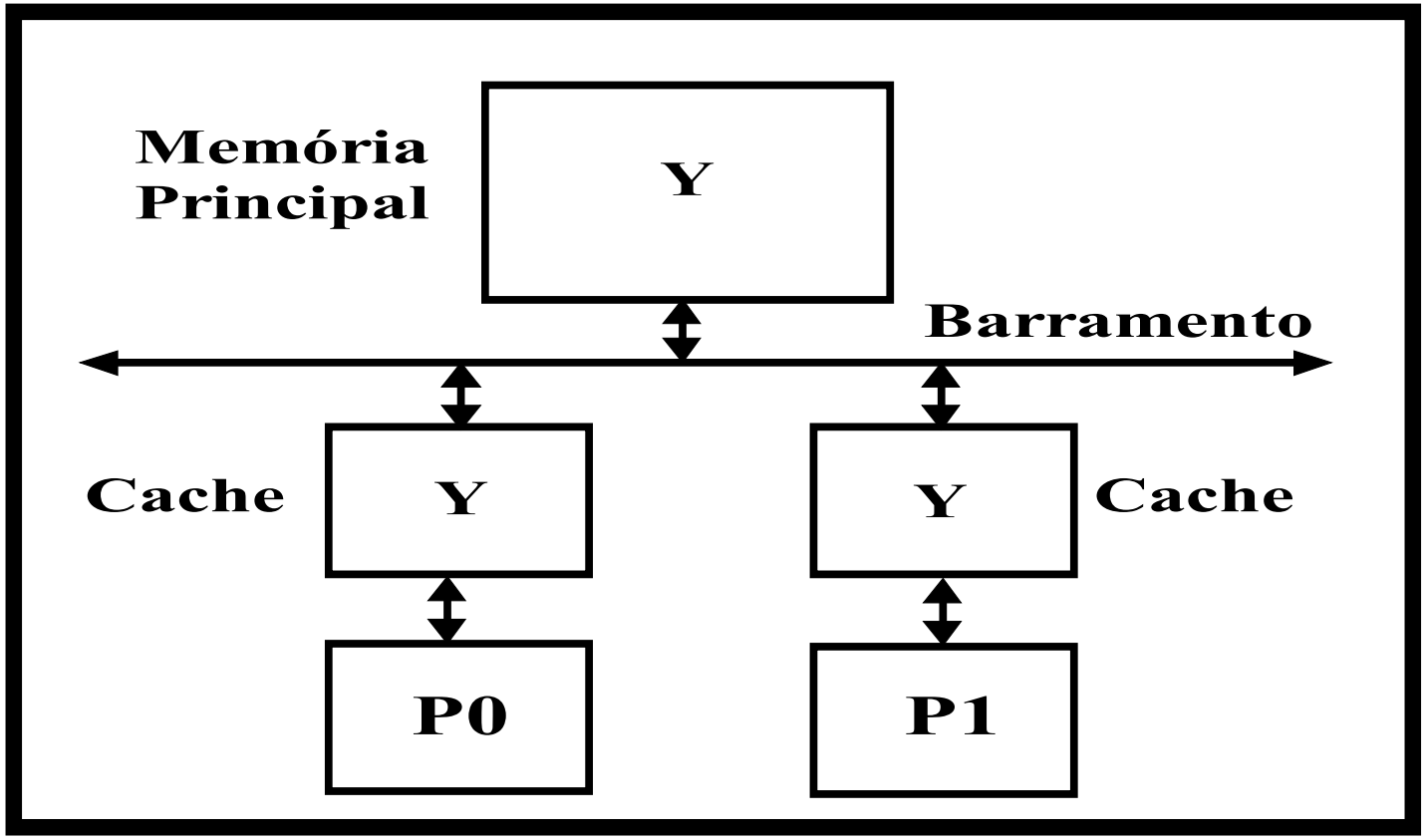
- A técnica de *"snooping"* para manutenção da coerência das memórias cache se baseia na existência de circuitos em cada nó de processamento que monitoram permanentemente o barramento de modo a invalidarem ou a atualizarem a cópia na memória cache local de um dado que esteja sendo alterado através de uma operação de escrita no barramento.
- Esta premissa é válida quando os caches operam no modo *"write through"*.

Coerência das Memórias Cache Mantida pela Técnica de *"Snooping"*



(a) Escrita com Invalidação

Coerência das Memórias Cache Mantida pela Técnica de *"Snooping"*



(b) Escrita com Atualização

Coerência das Memórias Cache Mantida pela Técnica de *"Snooping"*

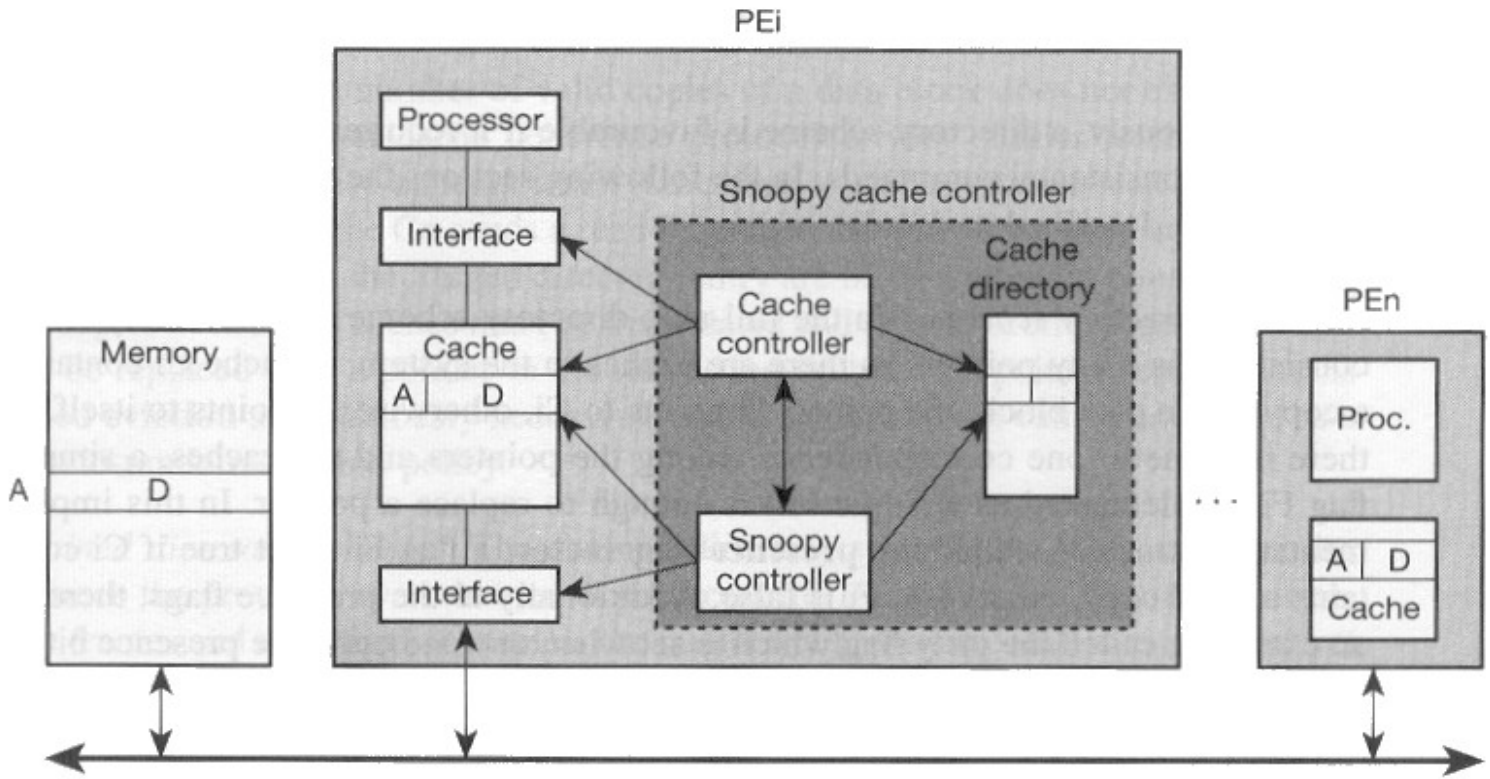


Figure 18.26 Structure of the snoopy cache controller.

Coerência de Cache

- Duas políticas podem ser adotadas em caso de falha no cache em operações de escrita:
  - *"write allocate"* → o bloco onde vai ser feita a operação de escrita é trazido primeiramente para a memória cache e a operação de escrita é então realizada na memória cache apenas, no modo *write back*, e também na memória principal, no modo *write through*.
  - *"no write allocate"* → o bloco a ser escrito não é trazido para a memória cache e, portanto, a operação de escrita sempre se realiza apenas na memória principal.

Coerência de Cache

- Usualmente a política *"write allocate"* é adotada com o modo *"write back"* e a política *"no write allocate"* com o modo *write through*.
- O esquema mais utilizado baseia-se numa combinação de *"write back"* com *"write invalidate"* e com a política *"write allocate"*.

Coerência de Cache

- Para o uso da técnica de *"snooping"* com o esquema *"write back"*, há a necessidade de se adotar protocolos mais complexos para a manutenção da coerência dos caches.
- Estes protocolos visam forçar a colocação no barramento de informações sobre operações de escrita sempre que necessário e a definição sobre quem deve fornecer um bloco de dados quando ocorre uma operação de leitura com falhas, já que no modo *"write back"*, a memória principal nem sempre possui uma cópia válida do bloco que está sendo solicitado.

Coerência de Cache

- **Caso uma das caches possua a informação atualizada (“*owner*”), enviará o dado solicitado, bloqueando a resposta da memória principal com a ativação de um sinal no barramento, já que o dado na memória pode estar desatualizado.**

Protocolo *MSI*

- Nesse protocolo, cada bloco na memória cache pode estar em um de 3 estados:
  - Inválido (I): bloco inválido na memória cache. É os estado inicial de todos os blocos da cache.
  - Shared (S) ou Compartilhado: o bloco só foi lido e pode haver cópias em outras memórias cache.
  - Modificado (M): Apenas uma das memórias cache possui uma cópia válida do bloco e a memória principal não está atualizada.
- Pressuposto: política de "write-allocate"

Protocolo *MSI*

Estado Atual	Tipo de Acesso	Próx. Estado	Ações no Barramento
Inválido (I)	Leitura Processador	Shared (S)	Leitura c/ falha
Shared (S)	Leitura Processador	Shared (S)	---
Shared (S)	Escrita Processador	Modificado(M)	Invalidação
Shared (S)	Leit. c/falha no barramento	Shared (S)	---
Shared (S)	Invalidação no Barramento	Inválido (I)	---
Modificado (M)	Leitura/Escr. Processador	Modificado(M)	---
Modificado (M)	Leitura c/ Falha no Barramento	Shared (S)	Aciona inibe *Envia bloco Atualiza mem.*

Protocolo *MSI*

Ação	Explicação
Leitura c/ falha	Leitura feita pelo controlador da cache, que faz um acesso à memória principal, trazendo o bloco correspondente ao endereço de leitura do processador para a memória cache local.
Invalidação	Operação especial feita pelo controlador da cache, que coloca o endereço correspondente ao bloco que está sendo escrito no barramento e ativando o sinal de invalidação, para avisar às demais caches que devem invalidar o bloco correspondente, caso esteja presente na sua cache local.
Envia bloco	O controlador da cache fornece o bloco para que seja atualizado nas demais caches,emulando a memória principal.

Protocolo *MESI*

- Nesse protocolo, cada bloco na memória cache pode estar em um de 4 estados:
  - Inválido (I): bloco inválido na memória cache
  - Shared (S) ou Compartilhado: bloco só foi lido e pode haver cópias em outras memórias cache
  - Modificado (M): apenas uma memória cache possui cópia do bloco e a memória principal não está atualizada
  - *Exclusivo (E):* apenas uma memória cache possui cópia do bloco e a memória principal está atualizada.
- Pressuposto: política de "write-allocate"

Estado Atual	Tipo de Acesso	Próx. Estado	Ações no Barramento
Inválido (I)	Leit. process. (*shared* desativado no barramento)	Exclusivo (E)	Leitura c/ falha
Inválido (I)	Leit. process. (*shared* ativado no barramento)	Shared (S)	Leitura c/ falha
Shared (S)	Leit. process.	Shared (S)	---
Shared (S)	escr. Process.	Modificado (M)	Invalidação
Shared (S)	Invalidação no barramento	Inválido (I)	---
Shared (S)	Leitura com falha no barramento	Shared (S)	Aciona *shared*
Exclusivo (E)	Leit. process.	Exclusivo (E)	---
Exclusivo (E)	Escr. process.	Modificado (M)	---
Exclusivo (E)	Leitura c/ falha no barramento	Shared (S)	Aciona *shared*
Modificado (M)	Leitura process.	Modificado (M)	---
Modificado (M)	Leitura c/ falha no barramento	Shared (S)	Aciona inibe Aciona *shared* *Envia bloco* *Atualiza mem.*

Protocolo *MOESI*

- **Nesse protocolo, cada bloco na memória cache pode estar em um de 5 estados:**
  - **Inválido (I): bloco inválido na memória cache.**
  - **Shared (S) ou Compartilhado: bloco só foi lido e pode haver cópias em outras memórias cache.**
  - **Modificado (M): apenas esse cache possui cópia do bloco e a memória principal não está atualizada**
  - ***Exclusivo (E):* Apenas esse cache possui cópia do bloco e a memória principal está atualizada.**
  - **Owner (O) : Esse cache supre o dado em caso de leitura com falha no barramento uma vez que a memória não está atualizada. Outros caches podem ter cópia do dado.**

Protocolo *MOESI*

Estado Atual	Tipo de Acesso	Próx. Estado	Ações no Barramento
Inválido (I)	leit. process. (*shared* desat. no barramento)	Exclusivo (E)	leitura c/ falha
Inválido (I)	leit. process. (*shared* ativado)	Shared (S)	leitura c/ falha
Shared (S)	leit. process.	Shared (S)	---
Shared (S)	escr. process.	Modificado (M)	invalidação
Shared (S)	invalidação no barramento	Inválido (I)	---
Shared (S)	leitura c/ falha no barramento	Shared (S)	aciona *shared*
Exclusivo (E)	leit. process.	Exclusivo (E)	---
Exclusivo (E)	escr. process.	Modificado (M)	---
Exclusivo (E)	leitura c/ falha no barramento	Shared (S)	aciona *shared*

Protocolo *MOESI*

Estado Atual	Tipo de Acesso	Próx. Estado	Ações no Barramento
Owner (O)	invalidação no barramento	Inválido (I)	---
Owner (O)	leitura c/ falha no barramento	Owner (O)	envia bloco aciona shared
Owner (O)	leit. process.	Owner (O)	
Owner (O)	escr. process.	Modificado (M)	---
Modificado (M)	leitura process.	Modificado (M)	---
Modificado (M)	leitura c/ falha no barramento	Owner (O)	*envia bloco* aciona inibe aciona *shared*

Protocolos de Coerência de Cache

- Nos três protocolos quando um bloco da memória cache no estado *Modificado* é selecionado para ser substituído para dar lugar a um bloco novo que está sendo lido da memória, o bloco antes de ser descartado deve ser atualizado na memória principal.
- No protocolo *MOESI*, este procedimento também deve ser adotado quando um bloco no estado *Owner* é selecionado para ser substituído.

Protocolos de Coerência de Cache com Diretório

Introdução

- Os esquemas de coerência até agora estudados baseiam-se fortemente no uso de operações de "broadcast" para invalidar os blocos em caches remotas
- O custo de operações de "broadcast" em redes multi-estágio é muito alto, tornando necessário a elaboração de esquemas alternativos
- Os esquemas de coerência baseados em diretório permitem que os comandos de consistência sejam enviados apenas para as caches que possuam cópia do bloco

Introdução

- A arquitetura básica utilizada pelos protocolos de coerência com diretório pressupõem uma memória compartilhada distribuída com caches locais e processadores em cada nó
- Cada módulo de memória possui um diretório local que armazena as informações sobre onde as cópias dos blocos estão residentes
- Os protocolos baseados em diretório enviam comandos de consistência seletivamente para aquelas caches que possuem uma cópia válida do bloco de dados compartilhado

Introdução

- Para realizar isso, uma entrada no diretório deve estar associada a cada bloco de dados.
- Cada entrada do diretório consiste de uma série de ponteiros para as caches que possuem uma cópia válida do bloco de dados.
- Uma forma compacta de armazenar essa informação é usar um bit para cada cache remota, que corresponderá a um ponteiro armazenado em um vetor único para cada módulo de memória.
- Um bit adicional indica ainda se algumas das caches com cópia válida pode atualizar o respectivo bloco de dados.

Estruturas de Diretórios

- Diretório totalmente mapeado
- Diretório Limitado
- Diretório Encadeado

- As métricas a serem consideradas na comparação desses esquemas são:
  - Quantidade de memória necessária para manter os diretórios
  - Número de comandos de consistência dos protocolos

Diretório Totalmente Mapeado

- Cada entrada no diretório possui tantos ponteiros quantas forem as caches no sistema
- Existe um bit de presença correspondente a cada processador, que indica se o bloco está presente ou não na sua cache
- O bit "dirty" indica se alguma das caches possui o bloco com exclusividade e pode atualizá-lo.
- Tamanho do diretório: N * M bits
  - N → número de processadores
  - M → número de blocos

Diretório Limitado

- Cada entrada no diretório possui um número (k) limitado de ponteiros para as caches do sistema
- Duas alternativas são possíveis:
  - Quando um processador k+1 requer uma cópia do bloco no seu cache, um dos processadores tem sua cópia invalidada e é substituído pelo novo processador
  - Podem co-existir mais de k cópias do bloco no estado "shared". Porém quando ocorre uma solicitação de escrita no bloco, uma mensagem de invalidação é enviada em "broadcast" para todos os processadores.
- Tamanho do diretório: $k * \log_2 N * M$ bits

Diretório Encadeado

- Cada entrada no diretório necessita possuir apenas uma posição vaga. A extensão dinâmica do número de ponteiros é conseguida distribuindo-se os ponteiros adicionais entre as diversas caches.
- Quando o primeira cache C_1 lê um bloco, o diretório cria um ponteiro para C_1 que recebe um ponteiro nulo
- Quando a cache C_2 realiza uma leitura subsequente do bloco, o diretório passa a apontar para C_2 e envia para C_2 um ponteiro para C_1.
- Em caso de invalidação uma mensagem é enviada para todos os membros da lista até que um ponteiro nulo seja encontrado.
- É difícil fazer a substituição de um bloco em uma cache no meio da lista → lista duplamente encadeada.

Estados dos Blocos da Cache

- O estado de cada cópia do bloco de dados na cache pode ser:
  - **Uncached**
    - Nenhum processador tem uma cópia desse bloco na cache
  - **Shared**
    - Um ou mais processadores possuem o bloco em seus caches e o bloco na memória está atualizado
  - **Exclusive**
    - Exatamente um processador tem uma cópia do bloco em seu cache e o bloco na memória está desatualizado. O processador é o "owner" do bloco

Coerência de Caches com Diretório

- **Definições:**
  - **Nó Requisitante**
    - **É o nó onde um pedido tem origem.**
  - **Nó residente**
    - **É o nó onde a posição de memória e a entrada do diretório residem.**
  - **Nó remoto**
    - **É o nó que tem uma cópia do bloco da cache, seja exclusivo ou compartilhado.**

Coerência de Caches com Diretório

- Estado Uncached
  - Read Miss
    - Bloco é enviado para o nó requisitante;
    - Nó requisitante é incluído na lista de ponteiros;
    - Estado do bloco muda para SHARED.
  - Write Miss
    - Bloco é enviado para o nó requisitante, que se torna "owner";
    - Nó requisitante é adicionado à lista de ponteiros;
    - Estado do bloco muda para EXCLUSIVE.

Coerência de Caches com Diretório

- Estado SHARED
  - Read Miss
    - Bloco é enviado para o nó requisitante;
    - Nó requisitante é incluído na lista de ponteiros.
  - Write Miss
    - Bloco é enviado para o nó requisitante, que se torna "owner";
    - Invalidação é envidada para todas as caches da lista de ponteiros;
    - Nó requisitante torna-se o único na lista de ponteiros;
    - Estado do bloco muda para EXCLUSIVE.

Coerência de Caches com Diretório

- Estado SHARED
  - Write Hit
    - O nó residente manda mensagem de invalidação para todos os nós diferentes daquele que está escrevendo;
    - Torna o nó que está escrevendo “owner”;
    - Estado do bloco muda para EXCLUSIVE.

Coerência de Caches com Diretório

- **Estado EXCLUSIVE**
  - **Read Miss**
    - Bloco é copiado do "owner" para a memória do nó residente;
    - Bloco é enviado ao nó requisitante;
    - Nó requisitante é incluído na lista de ponteiros;
    - Estado do bloco muda para "SHARED".
  - **Write-back** (Substituição do Bloco no "owner")
    - O bloco é copiado do "owner" para a memória do nó residente;
    - A lista de ponteiros fica vazia;
    - O estado do bloco muda para UNCACHED.

Coerência de Caches com Diretório

- Estado EXCLUSIVE
  - Write Miss
    - O bloco é invalidado no "owner" e transferido para o nó requisitante (novo "owner");
    - Novo "owner" torna-se único membro da lista de ponteiros.

Esquemas Alternativos

Coarse-Vector

- Também possui o número de entradas $i < N$;
- Quando o número de caches compartilhando o bloco é igual ou menor do que i, funciona como o esquema limitado;
- No caso contrário, a semântica da informação armazenada nessa entrada do diretório muda para indicar "coarse vector". Cada bit do vetor representa um grupo de processadores onde $r = N/(i*\log N)$;
- Um determinado bit é ativado se pelo menos um dos processadores do grupo correspondente contiver cópia do bloco de memória na sua cache.

Esquemas Alternativos

- LimitLESS
  - Proposto para uso na máquina Alewife do MIT;
  - Também possui o número de entradas i < N;
  - Quando ocorre que o número de caches compartilhando um bloco é maior que i, o processador do nó contendo a entrada do diretório é interrompido e copia os ponteiros para uma tabela "hash" em memória, liberando os ponteiros de "hardware" para uso;
  - Após a ocorrência de "overflow", a primeira escrita nesse bloco deve causar um "trap" , de modo a permitir a identificação dos ponteiros armazenados em memória para se fazer o envio das mensagens de invalidação;
  - Após a escrita, a entrada do diretório fica com o bit "dirty" ativado e os ponteiros em memória são liberados.

Esquemas Alternativos

- **Diretórios Esparsos**
  - O diretório é uma memória cache com menos entradas do que o número de blocos de memória existentes.
  - No caso de colisão, mensagens de invalidação são enviadas para todos os processadores presentes na entrada a ser substituída.
  - Após a invalidação, a entrada em conflito pode ser utilizada para armazenar a informação sobre o novo bloco compartilhado.
  - O problema óbvio deste esquema é a possibilidade de haver um número excessivo de mensagens de invalidação em virtude de conflitos nessa cache de diretório.

SCI (Scalable Coherent Interface)

SCI (Scalabe Coherent Interface)

- Padrão IEEE 1596-1992
- Define modelo de coerência de cache baseado em diretórios distribuídos, supondo um sistema com grande número de nós interligados através de interfaces SCI a uma rede
- Bloco na memória contém ponteiro para o primeiro elemento de uma lista duplamente encadeada de ponteiros para todos os nós que compartilham o bloco. Os ponteiros "forward" e "backward" são armazenados junto ao bloco de cache correspondente em cada nó
- Vantagem do modelo SCI de diretório distribuído: a memória ocupada com o diretório cresce naturalmente com a adição de outros nós com cache

SCI (Scalabe Coherent Interface)

- Desvantagem do modelo SCI de diretório distribuído: aumento da latência e da complexidade de operações de acesso à memória devido às operações adicionais para atualização das listas encadeadas
  - Exemplo: Nó A escreve no bloco X de memória compartilhado por outros N nós
    1. Nó A se desconecta da lista duplamente encadeada
    2. Nó A obtém elemento inicial da lista a partir da memória
    3. Nó A se coloca como elemento inicial da lista
    4. Nó A retira sequencialmente os outros N nós da lista, enviando para cada um deles uma ordem de invalidação
- Usa um protocolo do tipo "request-response" estrito → aumenta a latência, mas facilita a recuperação de erros