Hierarquia de Memória - Parte 5 - Introdução a Coerência de Cache

Arquitetura de Computadores

Emilio Francesquini
e.francesquini@ufabc.edu.br

2020.Q1

Centro de Matemática, Computação e Cognição
Universidade Federal do ABC

Disclaimer

UFABC

- Estes slides foram preparados para o curso de Arquitetura de Computadores na UFABC.
- Este material pode ser usado livremente desde que sejam mantidos, além deste aviso, os créditos aos autores e instituições.
- O conteúdo destes slides foi baseado no conteúdo do livro *Computer Organization And Design: The Hardware/Software Interface*, 5th Edition.

Paralelismo e hierarquia de memória: Coerência de Cache

O Problema da coerência de cache

UFABC

- Suponha que tenhamos 2 CPUs que compartilham o mesmo endereçamento físico
  - Têm caches write-through

Time step	Event	Cache contents for CPU A	Cache contents for CPU B	Memory contents for location X
0				0
1	CPU A reads X	0		0
2	CPU B reads X	0	0	0
3	CPU A stores 1 into X	1	0	1

O que é a coerência de cache

UFABC

- Informalmente, dizemos que a cache é coerente se leituras em um endereço devolvem o valor escrito mais recentemente.
- De maneira mais formal, a cache é coerente se ela satisfizer as seguintes propriedades:
  - Regra 1: Leituras após escritas
  - Regra 2: Propagação de modificações
  - Regra 3: Serialização das escritas

Regra 1: Leituras após escritas

UFABC

- Dados:
  - Um processador *P* escreve *V* na posição *X*
  - Não houve nenhuma escrita nesta posição por qualquer outro processador
- Então:
  - Uma leitura posterior feita por *P* em *X* devolve *V*

Regra 2: Propagação de modificações

UFABC

- Dados:
  - P_1 escreve V em X
- Então:
  - Após algum tempo e automaticamente, quando P_2 ler X deve obter V
- Por exemplo: Se a CPU B ler X após o passo 3 no exemplo anterior.

Regra 3: Serialização das escritas

UFABC

- Dados:
  - P_1 escreve V_0 em X e;
  - Em seguida, P_2 escreve V_1 em X
- Então:
  - Todos os processadores veem as escritas na mesma
- Isto é necessário pois, de outra maneira as caches de cada processador poderiam terminar com valores diferentes em X.

UFABC

- Protocolos de coerência de cache (*cache coherence protocols*) são operações feitas pelas caches dos multiprocessadores para garantir que as caches sejam coerentes.
- Duas operações principais são utilizadas:
  - Migração de dados para as caches locais
    - Reduz a banda para a memória compartilhada
  - Replicação de dados de leitura compartilhada
    - Reduz a contenção para o acesso

UFABC

Há dois tipos principais de protocolos de coerência de cache:

- Snooping Protocols
  - Cada cache monitora o bus em olhando todos as leituras e escritas
  - Implementação mais simples, porém menos escalável
- Protocolos baseados em diretório
  - Caches e memória armazenam o status de compartilhamento das linhas entre as caches em uma estrutura chamada de diretório
  - Protocolo mais escalável, porém mais complexo e com um overhead mais alto

- Caches tem pelo menos (no protocolo chamado MESI) 2 bits para flags a mais em cada uma de suas entradas que armazenam um dos 4 estados:
  - Modificada (suja) - o bloco foi modificado (está diferente da memória principal) e só está disponível nesta cache
  - Exclusiva - O bloco não está sujo e não há nenhuma cópia em qualquer outra cache
  - Compartilhada - O bloco é o mesmo da memória principal (não foi modificado) e pode estar presente em outra cache
  - Inválida - A entrada não contém dados válidos

UFABC

- Neste protocolo, as caches obtém acesso exclusivo a um bloco quando desejam escrever
  - Fazem um broadcast no bus avisando a todas as outras caches para invalidarem suas linhas
  - Leituras subsequentes à mesma linha pelas outras caches vão causar um miss
    - A cache que tiver o dado atualizado o fornece para as demais caches (já que elas estão monitorando o bus)

	M Modificada	E Exclusiva	S (*Shared*) Compartilhada	I Inválida
Linha válida?	Sim	Sim	Sim	Não
A RAM está...	desatualizada	válida	válida	-
Há cópias em outras caches	Não	Não	Talvez	Talvez
Uma escrita nessa linha	Fica na cache	Fica na cache	Vai pro barramento e atualiza a cache	Vai diretamente para o barramento

Exemplo

Processor activity	Bus activity	Contents of CPU A's cache	Contents of CPU B's cache	Contents of memory location X
				0
CPU A reads X	Cache miss for X	0		0
CPU B reads X	Cache miss for X	0	0	0
CPU A writes a 1 to X	Invalidation for X	1		0
CPU B reads X	Cache miss for X	1	1	1

- Mais informações veja a seção 17.3 do [WS].

Consistência da memória

UFABC

- Os modelos de consistência da memória definem quando as escritas passam a ser vistas pelos outros processadores
  - ”Ser vista”significa que leituras passam a devolver os valores mais recentemente escritos por outros processadores
  - É inviável que isso seja feito de maneira instantânea
- Algumas premissas:
  - Uma escrita é considerada como completa apenas quando todos os outros processadores já a viram
  - Um processador não reordena operações de escrita com outras operações de acesso à memória
- Consequências:
  - Se P escreve V_0 em X e depois escreve V_1 em Y
  - Então todos os processadores que conseguem ver Y com o valor $V1$ também conseguem ver X com o valor V_0
  - Processadores podem reordenar leituras, mas não escritas

Conclusões

Pegadinhas 1

- Endereçamento por bytes ou por palavras
  - Exemplo: cache com 32 bytes, mapeamento direto, blocos de 4 bytes
    - Byte 36 é mapeado para o bloco 1
    - Palavra 36 é mapeada para o bloco 4
- Ignorância dos efeitos do sistema de memória quando escrever ou gerar código
  - Exemplo: fazer a iteração de laços por colunas e não por linhas
  - Saltos muito longos nos vetores resultam em baixa localidade

Pegadinhas 2

UFABC

- Em um multiprocessador com L2 ou L3 compartilhada
  - Menos associatividade do que cores resulta em misses por conflito
  - Quanto mais cores mais associatividade é necessária
- Usar AMAT para avaliar o desempenho de processadores fora de ordem
  - Ignora o efeito de acessos não bloqueantes
  - Em vez disto, o ideal é avaliar o desempenho usando simulação

Conclusões

UFABC

- Memórias rápidas são pequenas, memórias grandes são lentas
  - Gostaríamos de ter memórias rápidas e grandes 😢
  - Caches nos dão essa ilusão 🙂
- Princípio da localidade
  - Programas usam uma pequena parte do seu espaço de memória com mais frequência
- Hierarquia de memória
  - Cache L1 $\leftrightarrow$ Cache L2 $\leftrightarrow$ ... $\leftrightarrow$ DRAM $\leftrightarrow$ Disco
- O design do sistema de memória é crítico para multiprocessadores.