Universidade Federal de São João del Rei - UFSJ.
Departamento de Ciência da Computação - DCOMP.
Arquitetura e Organização de Computadores - AOC II.

Relatório de análise dos resultados das implementações dos sistemas de cache inclusivo e exclusivo.

Bruno Lasmar, Daniel Marques, Lucas Ferreira e Rian Wagner

1 Introdução:

Em sistemas computacionais modernos, a velocidade do processador é muito maior do que a velocidade da memória principal (RAM). Isso cria um desequilíbrio conhecido como "lacuna de desempenho". Para mitigar esse problema, é comum o uso de hierarquias de memória, sendo o cache um componente essencial nessa estrutura.

O cache é uma memória de acesso rápido e de tamanho menor em comparação com a memória principal. Ele armazena cópias dos dados frequentemente acessados pela CPU, reduzindo a latência de acesso à memória principal. O objetivo principal do cache é melhorar o desempenho do sistema, minimizando o tempo de acesso à memória e aumentando a taxa de acertos de acesso aos dados.

No entanto, projetar um sistema de cache eficiente é um desafio complexo. Existem várias estratégias de organização, mapeamento e políticas de substituição que podem ser adotadas. Duas abordagens comuns são os sistemas de cache inclusivo e exclusivo.

2 Cache Inclusiva:

A cache inclusiva é um tipo de organização de cache em que os dados também estão presentes em caches superiores (como a L2) além da cache imediata (L1). Isso significa que cada bloco de dados na cache L1 também estará presente na cache L2.

Quando ocorre um acesso à memória, a cache L1 é a primeira a ser consultada. Se o dado estiver presente na cache L1 (cache hit), ocorre uma leitura ou gravação direta. Caso contrário, ocorre um cache miss e a cache L2 é consultada. Se o dado estiver presente na cache L2, ele é transferido para a cache L1 antes de ser lido ou escrito. Se o dado não estiver presente na cache L2, ocorre um cache miss em ambas as caches e o dado precisa ser buscado na memória principal.

A vantagem da cache inclusiva é que ela oferece uma maior taxa de acertos na cache L1, pois todos os dados da cache L1 também estão presentes na cache L2. Isso reduz o número de cache misses na cache L1 e melhora o desempenho geral do sistema.

No entanto, a desvantagem é que a cache inclusiva requer mais espaço de armazenamento, já que os mesmos dados estão presentes em múltiplas caches. Além disso, a atualização de dados em caches inclusivos pode ser mais complexa, pois as alterações precisam ser propagadas para todas as caches que contém o dado.

2 Cache Exclusiva:

A cache exclusiva é um tipo de organização de cache em que cada bloco de dados está presente em apenas uma cache específica, como a cache L1. Isso significa que os dados armazenados na cache L1 não estão duplicados em nenhuma outra cache.

Quando ocorre um acesso à memória, a cache L1 é a primeira a ser consultada. Se o dado estiver presente na cache L1 (cache hit), ocorre uma leitura ou gravação direta. Caso contrário, ocorre um cache miss e o dado precisa ser buscado na memória principal. Se a cache L1 estiver cheia, é necessário substituir um bloco de dados existente para acomodar o novo dado.

Ao contrário da cache inclusiva, na cache exclusiva não há uma cache L2 que contenha os mesmos dados que estão presentes na cache L1. Isso significa que um cache miss na cache L1 não é verificado na cache L2. Isso reduz o espaço necessário para o armazenamento de dados, mas também aumenta a probabilidade de ocorrerem cache misses.

3 Implementação:

1º) Criação dos Vetores: Antes de iniciar a implementação do algoritmo de cache, é necessário criar os vetores de dados que serão utilizados no processo. Isso é feito por meio de função, que recebe o tamanho dos vetores A e B e retorna os vetores preenchidos com os dados correspondentes.

2º) Inicialização da Cache e RAM: Após a criação dos vetores, é feita a inicialização da memória cache (Cache) e da RAM principal (RAM) com os parâmetros adequados, como tamanho da palavra, tamanho da linha e tamanho total da cache.

3º) Implementação do Algoritmo de Cache: A implementação do algoritmo de cache é realizada em um loop aninhado, onde cada iteração representa o acesso a um elemento do vetor A. Dentro desse loop, é feito o acesso aos elementos dos vetores A e B, e a cache é atualizada de acordo com o estado de cada elemento.

4 Resultados:

O programa foi desenvolvido no ambiente Windows utilizando python 3.11.4, os resultados foram computados para cada tamanho dos arrays:

Tamanho	Exclusivo	Inclusivo
4	Total de Cache Hit: 27 Hit em L1: 27 Hit em L2: 0 Total de Cache Miss: 10 Miss em L1: 5 Miss em L2: 5	Total de Cache Hit: 27 Hit em L1: 27 Hit em L2: 0 Total de Cache Miss: 10 Miss em L1: 5 Miss em L2: 5
8	Total de Cache Hit: 110 Hit em L1: 110 Hit em L2: 0	Total de Cache Hit: 110 Hit em L1: 110 Hit em L2: 0

	Total de Cache Miss: 36 Miss em L1: 18 Miss em L2: 18	Total de Cache Miss: 36 Miss em L1: 18 Miss em L2: 18
16	Total de Cache Hit: 444 Hit em L1: 444 Hit em L2: 0 Total de Cache Miss: 136 Miss em L1: 68 Miss em L2: 68	Total de Cache Hit: 444 Hit em L1: 444 Hit em L2: 0 Total de Cache Miss: 136 Miss em L1: 68 Miss em L2: 68
128	Total de Cache Hit: 28640 Hit em L1: 25056 Hit em L2: 3584 Total de Cache Miss: 11840 Miss em L1: 7712 Miss em L2: 4128	Total de Cache Hit: 28640 Hit em L1: 25056 Hit em L2: 3584 Total de Cache Miss: 11840 Miss em L1: 7712 Miss em L2: 4128
512	Total de Cache Hit: 458624 Hit em L1: 229248 Hit em L2: 229376 Total de Cache Miss: 360704 Miss em L1: 295040 Miss em L2: 65664	Total de Cache Hit: 458624 Hit em L1: 229248 Hit em L2: 229376 Total de Cache Miss: 360704 Miss em L1: 295040 Miss em L2: 65664

Para que as taxas de erro sejam significativas e diferentes de um sistema de cache para o outro é necessário um número muito grande, como o array de 2048. No entanto, o algoritmo tomaria complexidade de tempo exponencial. Para poder visualizar de fato como aos sistemas funcionam de forma diferente, reduzimos a cache e o vetor proporcionalmente:

Cache	Exclusiva	Inclusiva
L1 = 40 L2 = 640 Vetor = 20	Total de Cache Hit: 685 Hit em L1: 300 Hit em L2: 385 Total de Cache Miss: 615 Miss em L1: 500 Miss em L2: 115	Total de Cache Hit: 689 Hit em L1: 300 Hit em L2: 389 Total de Cache Miss: 611 Miss em L1: 500 Miss em L2: 111

Respondendo aos questionamentos propostos:

1. Houve diferença entre as taxas de erros nos esquemas inclusivo e exclusivo de cache? Compare a taxa de erros somente na L1 e erros nos dois níveis de cache para os dois sistemas simulados

Houve diferença. Comparando somente a L1 não percebemos diferença. No entanto, ao compararmos os dois sistemas percebemos uma diferença de erros na L2 exclusiva, se compararmos com a quantidade de erros na L2 inclusiva.

2. No esquema exclusivo, quantas referências resultaram na utilização da cache L2 como cache de vítima (alocando um bloco removido da cache L1)?

Todas as referências utilizaram L2 como cache de vítimas, uma vez que ela é suficientemente grande para alocar todas as referências.

5 Conclusão:

A modelagem e implementação do algoritmo de cache permitem otimizar o acesso aos dados armazenados na RAM principal, reduzindo a latência e melhorando o desempenho geral do sistema. O algoritmo de cache implementado utiliza estruturas de classes e métodos para gerenciar a cache, realizar buscas inclusivas e exclusivas, e atualizar os dados conforme necessário. A criação dos vetores de dados iniciais e a correta inicialização da cache e RAM são etapas importantes para garantir o correto funcionamento do algoritmo.

Em resumo, a modelagem e implementação desse algoritmo de cache em Python fornecem uma solução eficiente para melhorar o desempenho do acesso à memória principal em sistemas computacionais.