ESTUDO DE TÉCNICAS PARA REDUÇÃO DA TAXA DE FALTAS NA MEMÓRIA CACHE

ANDRÉ MARCELO COELHO DA SILVA[1]; LUCIANO VOLCAN AGOSTINI[2]

[1]*Universidade Federal de Pelotas – amcdsilva@inf.ufpel.edu.br*
[2]*Universidade Federal de Pelotas – agostini@inf.ufpel.edu.br*

1. INTRODUÇÃO

O desempenho do processador é fator determinante para a performance dos sistemas computacionais e, portanto, maiores ganhos de desempenho são necessários, pois, as exigências dos softwares estão sempre em crescimento. Um elemento de hardware essencial para esse desempenho e que trabalha em conjunto ao processador, é a memória cache (UFRPE, 2020). Esse tipo de memória é mais rápido do que a memória externa e permite que o processador fique menos ocioso quando necessita de dados da memória. Atualmente, todos os processadores de alto desempenho trazem uma certa quantidade de memória cache embutida no seu encapsulamento e quanto maior o espaço dedicado para as memórias caches, maiores os desempenhos atingidos.
Projetar memórias caches eficientes não é algo trivial, pois existem vários fatores que devem ser observados de forma conjunta no momento do projeto, a fim de estabelecer a melhor configuração possível. Para que a cache seja eficiente, é importante que, quando o processador requisitar um dado de memória, esse dado esteja na cache, evitando a busca deste dado na memória externa. O número de vezes em que o dado necessário não está na cache é chamado de taxa de faltas (PATTERSON, 2000) e esse número precisa ser reduzido ao máximo, para ampliar a eficiência da cache.
Nesse sentido, este trabalho apresenta resultados obtidos com a realização de experimentos com diversas configurações da memória cache, com o intuito de reduzir a taxa de faltas e, consequentemente, aumentar o desempenho do processador.

2. METODOLOGIA

Relatos na literatura (PATTERSON, 2000) indicam que o tempo gasto com leituras e escritas na memória é uma parcela significativa do tempo total de processamento. Em um projeto de memórias caches, existem alguns fatores que influenciam no desempenho final, dentre eles estão o tempo de acerto, a taxa de faltas e a penalidade de faltas.
Assim, foi realizado um estudo inicial do comportamento das memórias caches com diferentes configurações. As implementações e os valores foram extraídos na ferramenta simplescalar (SIMPLESCALAR, 2020). Para realizar os experimentos com valores realistas de implementação e, assim, fazer comparações justas de desempenho, foi utilizado o benchmark gcc1 (PATTERSON, 2000) em todos os casos.
Neste artigo foram descritos os resultados dos experimentos alterando as seguintes características: (i) tamanho do bloco, (ii) aumento do tamanho da memória cache, (iii) aumento da associatividade, e (iv) caches multiníveis. Estes critérios estão detalhados em (TANENBAUM, 2007). Cada item, terá os resultados obtidos comentados em tópicos distintos na sessão nos resultados e

discussão. Por fim, serão discutidos os resultados obtidos e apresentadas as propostas para trabalhos futuros.

3. RESULTADOS E DISCUSSÃO

Otimização 1: Aumento do tamanho do bloco

Nessa proposta de configuração, foram utilizados quatro tamanhos de blocos (16, 32, 64 e 128 bytes), utilizando o mapeamento direto e o conjunto associativo de quatro vias (TANENBAUM, 2007), respectivamente, com o objetivo de verificar os efeitos do princípio da localidade espacial e também a redução do número de faltas compulsórias, que são faltas por não existirem dados na memória cache na inicialização do sistema. Os resultados estão apresentados na Figura 1.

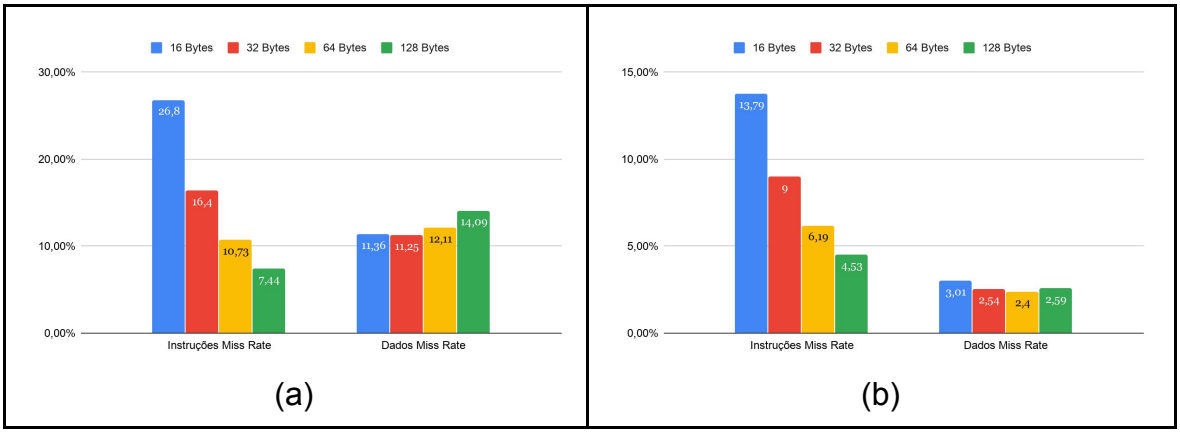

Figura 1 - Taxa de faltas para diversos tamanhos de blocos utilizando (a) o mapeamento direto e (b) mapeamento associativo por conjunto de quatro vias

Para a cache de instruções, fica evidente a redução nas taxas de falhas, nos dois cenários da Figura 1, quando comparamos o tamanho dos blocos utilizados. Essa redução demonstra que, neste tipo de memória, deve-se utilizar blocos de tamanhos maiores, visto que ela explora melhor a localidade espacial.

Na cache de dados, não acontece a mesma queda nos valores. Com mapeamento direto existe uma pequena redução entre os valores de 16 e 32 bytes mas, após, os índices começam a subir. Isso indica que o aumento do tamanho do bloco não funciona sempre e que existe um ponto limite. Essa inflexão na curva acontece devido a redução na exploração da localidade temporal. Assim, a memória de dados tem maior eficiência com blocos de tamanhos intermediários, pois nessa configuração, a localidade temporal é melhor explorada.

Otimização 2: Aumento de capacidade da memória cache

Para esse experimento, foi usado o mapeamento direto, com blocos de tamanhos fixos de 32 bytes, aumentando o número de conjuntos da memória (TANENBAUM, 2007). Assim, os tamanhos das quatro memórias caches simuladas foram de 2, 4, 8 e 16 Kbytes, respectivamente. Com aumento da capacidade, houve redução na taxa de faltas em ambas as memórias. Isto se explica pois, quanto maior for o tamanho da memória, maior será o número de blocos que ela suporta e a probabilidade de uma informação requisitada estar presente também aumenta. Consequentemente, há uma redução no número de faltas. A desvantagem que essa configuração apresenta é de elevar o tempo de

acerto, pois as caches maiores, demoram um tempo maior para dar a resposta, já que devem percorrer um caminho interno maior.

Otimização 3: Aumento da associatividade

Nessa configuração, foi utilizada uma memória de 8 Kbytes de tamanho total. Houve redução nas faltas em todos os tipos de mapeamentos implementados, tanto para instruções quanto para dados. Para a cache de instruções, os resultados indicam que a associativa de quatro vias apresenta o menor resultado, sendo também o ponto de inversão de tendência, visto que uma associatividade de oito vias apresentou resultados piores. Contudo, a memória de dados não apresentou ponto de inflexão de tendência nos testes realizados, indicando que a memória totalmente associativa é a que apresenta a menor taxa de faltas. Mais uma vez, os resultados demonstram que as memórias de instruções e dados se comportam de modos diferentes. Os resultados estão apresentados na Figura 2.

Como desvantagem, o aumento da associatividade apresenta um tempo de acerto maior, devido à componente adicional no final do circuito para realizar a escolha de qual é a palavra correta dentro do conjunto. Neste trabalho foram utilizados conjuntos com 2, 4, 8 e 16 elementos.

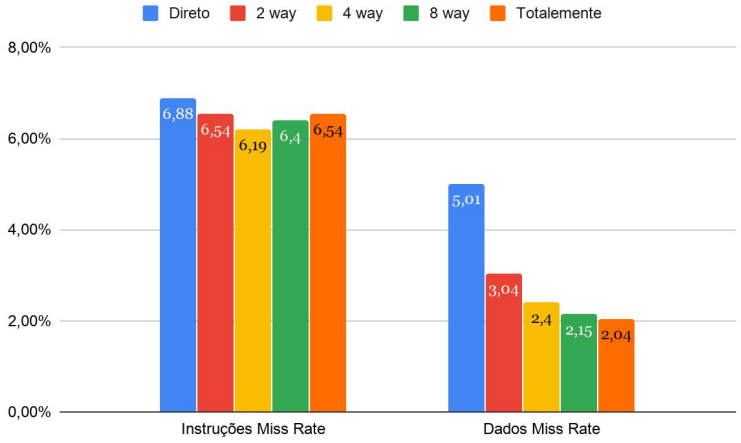

Figura 2 - Taxa de faltas com aumento da associatividade

Otimização 4: Caches multiníveis

Nesse experimento, a primeira configuração considera caches separadas para o nível L1 e caches unificadas para o nível L2 (TANENBAUM, 2007). A segunda, utilizada caches separadas para os dois níveis. Os tamanhos das memórias utilizados foi de 4 Kbytes para cada, na versão separada, e de 32 Kbytes para a memória de segundo nível. Na implementação de L2 separada, foram utilizadas duas memórias de 16 Kbytes. Em todas as implementações foram utilizados blocos de 32 bytes e mapeamento direto para obter resultados justos. Estes valores indicam um aumento significativo na taxa de faltas quando comparadas L1 e L2. Esse resultado deve ser analisado por dois ângulos, a taxa de faltas local e global de L2. A taxa global se refere a percentagem de faltas total de L2, que na configuração experimentada apresenta um valor elevado (na unificada foi 35,8% e na separada foi 52,87% para instruções e 28,89% para dados). Essa percentagem é calculada sobre o número absoluto de faltas que aconteceram em L1.

Como já citado, com a inclusão de mais um nível de cache, a taxa de faltas não é alterada para L1. Os valores são os mesmos para ambas as configurações. A diferença está na taxa de faltas do nível 2 utilizando implementação de caches diferentes. Os resultados consideram o número total de instruções e de *loads e*

stores executadas e, também, a taxa de faltas apresentadas no nível 1. A partir desses dados, foi calculado o número de acessos que serão feitos ao nível 2. Já em L2, a percentagem de faltas teve valores distintos: para a cache unificada foi 35,8% e para a separada foi 52,87% para instruções e 28,89% para dados. A partir desses valores, foi calculado o valor total de faltas na memória em L2. A memória de segundo nível unificada apresenta, aproximadamente, 1 milhão a menos de faltas quando comparada à memória separada, portanto, se mostrando mais eficiente para ser utilizada em L2, quando consideramos somente o número de faltas.

4. CONCLUSÕES

Neste trabalho verificou-se que as caches de instruções e dados possuem comportamentos diferentes, já que os resultados na taxa de faltas foram sempre distintos para as mesma configurações. Além disso, o mapeamento conjunto associativa de quatro vias utilizando blocos de 128 Kbytes, foi a configuração que apresentou a menor taxa de faltas (4,53%) entre todas as configurações avaliadas para a cache de instruções. Por outro lado, a memória de dados totalmente associativa obteve melhor desempenho, com 2,04% de faltas. Isso mostra que as caches de dados têm uma tendência a explorar melhor a localidade temporal.
Os resultados indicam que a taxa de faltas no nível 2 é menor quando se utilizada uma memória unificada de dados e instruções. Esse resultado justifica-se devido à unificação ter um tamanho quatro vezes maior que as caches separadas somadas e, como indicam os resultados obtidos na otimização 2, memórias maiores possuem valores menores para a taxa de falta.
Outro ponto importante é a sensível redução na taxa de faltas quando se utiliza tamanhos de blocos maiores, principalmente na memória de instruções, pois ela explora muito bem a localidade temporal. Como era esperado, o aumento do tamanho memória cache faz reduzir a taxa de faltas, já que caches maiores podem armazenar uma quantidade maior de blocos. Como trabalhos futuros, pretende-se investigar os efeitos em memórias com maior capacidade, com tamanho do bloco maior e com outros graus de associatividade.

5. REFERÊNCIAS BIBLIOGRÁFICAS

PATTERSON, D.A. **Organização e projeto de computadores: A interface Hardware/Software**. Rio de Janeiro: LTC, 2000. 2v.

TANENBAUM, A.S. **Organização estruturada de computadores**. São Paulo: Pearson Hall, 2007. 5v.

SIMPLESCALAR. Acessado em 23 ago. 2020. Online. Disponível em: http://www.simplescalar.com/

UFRPE. Deinfo. Memória cache: Arquitetura e políticas de leitura e escrita. Acessado em 25 ago. 2020. Online. Disponível em: http://ww2.deinfo.ufrpe.br/sites/ww2.deinfo.ufrpe.br/files/artigos_aoc/Memoria%20cache.pdf