Análise de Desempenho de Caches de Dados e Instruções utilizando Simulador Simplescalar

Pedro Henrique Warken Ramos

[1]Universidade Federal de Santa Maria (UFSM)

1. Introdução

Este trabalho tem como objetivo investigar o desempenho de caches de dados e instruções por meio do uso do simulador Simplescalar, uma ferramenta amplamente utilizada para a análise e avaliação de arquiteturas de computadores. O estudo foi conduzido com a execução de três aplicações de testes providas pelo professor Mateus Beck Rutzig, visando explorar os conceitos apresentados na disciplina de Arquitetura de Computadores.

Foram modeladas quatro caches de nível 1, variando-se os parâmetros de configuração, tais como o número total de conjuntos, o tamanho do bloco em bytes, a associatividade e o algoritmo de substituição. Tais parâmetros foram testados por meio de linha de comando, usando o módulo "sim-cache"do SimpleScalar, as configurações das caches foram especificadas, permitindo a execução das aplicações selecionadas. Para este trabalho foi definido o tamanho padrão das caches como 1 kilobyte, para que somente a organização dos parâmetros influencie nos resultados e não o tamanho total da cache.

As aplicações utilizadas foram selecionadas com o intuito de abranger diferentes cenários de uso, contemplando tarefas como multiplicação de matrizes, decodificação de arquivos MPEG e cálculos matemáticos simples. Durante a execução das aplicações, foram coletados dados relevantes para análise, como o número de acessos às caches, o número de acertos (hits) e o número de faltas (misses).

A partir dos dados coletados, foi realizado uma análise comparativa do desempenho das quatro caches de dados e instruções. Essa análise foi realizada tanto quantitativamente, através da elaboração de gráficos com os resultados obtidos, quanto tecnicamente, argumentando sobre os resultados observados e relacionando-os com os conceitos apresentados na disciplina de Arquitetura de Computadores.

2. Metodologia

Nesta seção, será descrito em detalhes a metodologia utilizada para realizar a avaliação comparativa de desempenho das caches montadas no Simplescalar. Serão abordados os passos desde a configuração das caches até a coleta e análise dos resultados.

2.1. Configuração das Caches

Neste estudo, foi utilizado o simulador Simplescalar "sim-cache"para modelar e simular as caches. Foram configuradas quatro caches de dados e instruções de nível 1, variando os seguintes parâmetros: número total de conjuntos (<nsets>), tamanho do bloco em bytes (<bsize>), associatividade (<assoc>) e algoritmo de substituição (<repl>). Cada cache foi otimizada para uma aplicação específica, enquanto uma cache adicional foi projetada para fornecer a melhor otimização média.

As configurações das caches foram definidas da seguinte forma:

Cache 1 (Otimizada para a aplicação ”mm.ss”):

- Cache de instruções: il1:1:256:4:l
- Cache de dados: dl1:2:64:8:l

Cache 2 (Otimizada para a aplicação ”basicmath.ss”):

- Cache de instruções: il1:1:1024:1:l
- Cache de dados: dl1:4:32:8:l

Cache 3 (Otimizada para a aplicação ”amp.ss”):

- Cache de instruções: il1:2:256:2:l
- Cache de dados: dl1:1:32:32:l

Cache 4 (Melhor otimização média):

- Cache de instruções: il1:2:256:2:l
- Cache de dados: dl1:2:32:16:l

Cada configuração de cache foi projetada para atender às demandas específicas de uma determinada aplicação, considerando o tamanho dos conjuntos, o tamanho do bloco e a associatividade. Como o objetivo do trabalho é encontrar as caches mais otimizadas para cada aplicação o algoritmo de substituição utilizado foi o LRU (Least Recently Used) para todas as caches visto que tanto o FIFO (First In First Out) e o Random aproveitam de forma menos eficiente a localidade temporal das aplicações.

Para encontrar a cache de maior otimização média foi analisado quais atributos mais influenciavam no desempenho de cada uma das caches e foi usado o valor que era mais encontrado nas caches. Em seguida esses valores iniciais deduzidos de forma aproximada foram testados para verificar se realmente eram coerentes com a realidade até que foi encontrado a cache mais balanceada para as 3 aplicações.

3. Resultados

A análise comparativa do desempenho das quatro caches de dados e instruções foi realizada com base nos dados coletados durante a execução das aplicações de teste. Foram avaliados o número de acertos (hits) e o número de faltas (misses) tanto da cache de dados quanto a de instruções.

3.1. Gráficos

Nas Figuras 1, 2 e 3, são apresentados os resultados obtidos para cada aplicação específica: ”mm.ss”, ”basicmath.ss”e ”amp.ss”, respectivamente. Cada figura mostra a performance das quatro caches em relação aos hits e misses nas caches de instruções (il1) e de dados (dl1).

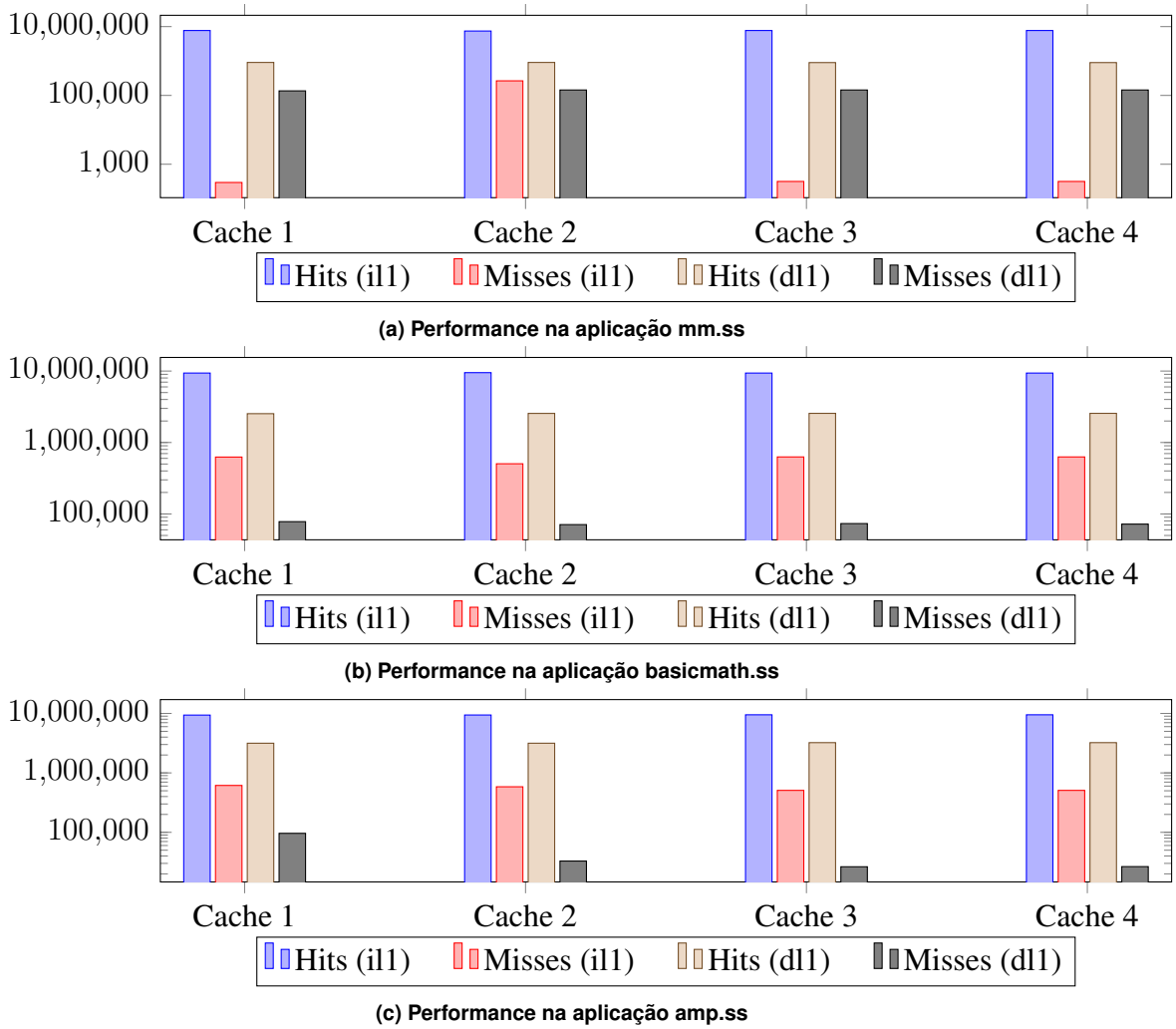

Figura 1. Comparação de performance das caches

3.2. Explicação dos resultados

Cache 1 - Otimizada para mm.ss: A multiplicação de matrizes geralmente envolve acessar elementos das matrizes de forma sequencial. Essa operação exibe uma boa localidade espacial, pois acessa locais de memória contíguos. Além disso, a multiplicação de matrizes envolve cálculos repetitivos nos mesmos dados, resultando em uma boa localidade temporal.

Dessa forma nessa cache foi usado o maior tamanho de bloco de todas aplicações em sua cache de dados, 64 bytes. Como a multiplicação de matrizes também envolve boa localidade temporal foi usado uma associatividade de 8 para acomodar essa necessidade. A multiplicação de matrizes envolve acessar elementos de diferentes linhas e colunas em um padrão específico. Ao armazenar mais elementos da mesma linha ou coluna juntos, uma maior associatividade permite uma melhor acomodação da localidade espacial presente na multiplicação de matrizes do que um maior número de conjuntos, portanto o número de conjuntos escolhido foi de apenas 2.

Embora a aplicação possa haver algumas declarações condicionais dentro dos loops, como verificações de limite ou condições de término, o número geral de ramificações

na multiplicação de matrizes é geralmente menor em comparação com aplicativos com fluxo de controle mais complexo. Dessa forma o tamanho de bloco mais efeitivo para a cache de instruções foi de 256 bytes, que explora bem a localidade espacial da aplicação mas ainda assim deixa espaço para para uma associatividade de 4, permitindo armazenar um número moderado de instruções de branch juntamente com seus respectivos destinos.

Cache 2 - Otimizada para basicmath.ss: Realizar cálculos matemáticos simples envolve manipular valores individuais ou pequenos conjuntos de dados. As operações de resolver funções cúbicas ou converter ângulos, não apresentam uma localidade espacial ou temporal forte. Os dados acessados podem estar dispersos pela memória, levando a uma menor localidade espacial. Além disso, como cada cálculo é independente, pode não haver uma localidade temporal significativa também. Portanto, basicmath.ss provavelmente terá uma localidade espacial e temporal menor em comparação com as outras aplicações para sua cache de dados.

Com isso foi usado um tamanho de bloco de 32 bytes para a cache de dados, um número menor do que na cache 1, e foi mais valorizado os aspectos de número de conjuntos e associatividade. Algumas operações matemáticas envolvem algoritmos iterativos ou recursivos, nos quais os resultados de cálculos anteriores são reutilizados nas etapas subsequentes. Com uma associatividade maior, os dados necessários para esses cálculos podem permanecer na cache por períodos mais longos, facilitando a reutilização eficiente de dados. Dessa forma a associatividade para a cache de dados da cache 2 foi de 8, para ajudar a evitar conflitos dentro dos conjuntos, e o número de conjuntos foi de 4 bytes, para ajudar a evitar contenções de índices.

Cálculos matemáticos simples, geralmente não contém muitas ramificações, é esperado que basicmath.ss tenha um número relativamente menor de ramificações em comparação com as outras aplicações. Dessa forma a maior performance foi observado usando uma cache de instruções com valor máximo possível, dentro do limite de 1 kilobyte, de tamanho de bloco. Tendo menos ramificações a aplicação itera quase que linearmente pelas instruções, o que gera uma grande localidade espacial, que fica bem aproveitada por grandes tamanhos de bloco.

Cache 3 - Otimizada para amp.ss: Decodificar um arquivo MPEG envolve ler e processar dados de áudio. Embora possa haver algum acesso sequencial aos dados, a decodificação de áudio também envolve acessar partes diferentes do arquivo de forma não linear devido às técnicas de compressão. Isso pode reduzir a localidade espacial em certa medida. No entanto, como os dados de áudio geralmente são processados em pequenos trechos, pode haver uma boa localidade temporal, pois os mesmos dados podem ser acessados várias vezes durante a decodificação. Em geral, amp.ss pode ter uma localidade espacial moderada, mas uma boa localidade temporal para sua cache de dados.

Tendo isso em vista a cache de dados escolhida para essa aplicação foi uma cache diretamente mapeada. Em uma cache de mapeamento direto, cada bloco de memória pode ser mapeado apenas para uma linha de cache específica. Isso simplifica o processo de busca na cache, pois o processador pode determinar diretamente a linha de cache em que um bloco de memória deve residir sem precisar pesquisar em vários conjuntos de cache.

A vantagem desse tipo de cache de dados para essa aplicação é explicado pelo fato

de que arquivos de áudio MPEG geralmente são lidos sequencialmente do início ao fim durante a decodificação. Uma cache de mapeamento direto é adequada para padrões de acesso sequencial como esse, pois permite o pré-carregamento eficiente e a recuperação de blocos de dados. Como os dados são acessados em uma ordem previsível, a cache de mapeamento direto garante que os dados necessários estejam disponíveis na cache para o processador buscar rapidamente, minimizando as falhas de cache. Como a localidade espacial é moderada a cache de dados mais performática para essa aplicação foi composta de um tamanho de bloco de 32 bytes e pela vantagem do mapeamento direto nesse tipo de aplicação foi usado todo espaço restante da cache para o número de conjuntos, que ficou com 32 bytes também.

A compressão MPEG usa várias técnicas, como codificação de comprimento variável, quantização e codificação de entropia, que podem levar a um grande número de ramificações. Além disso, a decodificação de áudio pode envolver tratamento de erros, sincronização e operações de busca que podem introduzir ramificações adicionais. Portanto foi utilizado uma cache de instruções com tamanho de bloco moderado de 256 bytes e o restante do espaço disponível foi usado para aumentar o número de conjuntos e de associatividade, criando uma cache associativa por conjunto de 2 vias, o que ajuda a evitar misses por conflitos de mapeamento que são mais comuns em uma aplicações com mais instruções de branch.

Cache 4 - Otimizada para melhor performance entre as aplicações: A quarta cache foi projetada para buscar a maior performance média entre as três aplicações consideradas: mm.ss, basicmath.ss e amp.ss. Uma abordagem equilibrada foi adotada levando em consideração as características de cada aplicação e buscando valorizar mais os atributos mais comuns entre as caches.

Dessa forma a cache de dados mais performática teve um tamanho de bloco de 32 bytes, tamanho usado tanto na cache 2 quanto na cache 3. Para manter um equilíbrio entre os atributos das caches foi escolhido um número de conjuntos de 2, que fica bem próximo dos valores definidos para as demais caches, e uma associatividade de 16, que fica em um meio termo entre as 3 caches mas ainda assim na média teve uma performance superior distribuída entre todas aplicações.

Para a cache de instruções foi usado um tamanho de bloco de 256 bytes adequado para a cache 1 e 3 mas que também não apresentou uma redução tão grande em performance para a cache 2. O número de conjuntos e de associatividade foi definido como 2, bem próximo dos demais valores das caches.