Fábio Augusto Menocci Cappabianco RA991724

Para melhorar o acesso à memória, modos de acesso tais como acesso paginado e read-modify-write têm sido idealizados.

Na maioria dos acessos à memória de sistemas específicos pesados, acessos a vetores são realizados. O artigo apresentou uma nova estratégia para minimizar os acessos da memória DRAM, por maximizar o número de acessos paginados, a partir de três princípios: determinar o número e tamanho de memórias, a maneira de atribuir ou colocar os vetores na memória e a seqüência de acesso dos vetores por operações de programas.

Um acesso normal à memória é realizado por um estágio de decodificação de linha, onde uma linha inteira contendo m palavras é copiada em um buffer de linha, seguindo de um estágio de decodificação de coluna ou seleção do elemento procurado na linha copiada. Posteriormente uma leitura ou escrita é realizada. Ao final do processo um novo estágio de decodificação de linha é iniciado.

Em um acesso paginado, depois de uma leitura/escrita completada, caso o próximo acesso seja na mesma linha, então somente o segundo estágio de decodificação de coluna é realizado, pois a linha desejada já está no buffer.Para se maximizar o acesso a páginas mais eficiente, dois passos são realizados: (1) alocar e mapear os vetores do código nos limites da memória. (2) escalonar os acessos no código.

O primeiro passo tenta encontrar uma configuração de memória melhor para o segundo paso, de modo que a latência de acesso à memória seja reduzida. O algoritmo citado trabalha a fim de encontrar a menor latência dentro de um determinado limite de memória. Quanto maior a instância de memória maior a latência, porém menos espaço é utilizado.

Inicialmente cada vetor é alocado em uma instância de memória diferente. Unem-se, então, pares de memórias que aumentem menos a latência, sequencialmente, até que o tamanho da memória utilizada seja menor que o limite dado. Vetores agrupados na mesma memória não podem ser acessados ao mesmo tempo na execução de instruções.

No segundo passo, faz-se um reagendamento das instruções que acessam vetores na memória, maximizando os acessos paginados na estrutura do passo anterior. Para resequenciar o código, são calculadas a latência inicial e a latência após a realocação de cada operação individualmente em cada posição possível do código, que não altere a dependência de dados. A alteração que produz o maior número de acessos paginados é escolhida, fixando-se a posição da operação correspondente. Segue-se fazendo alterações até que todas as operações sejam fixadas. A subseqüência de alterações do total realizado que possui o maior número de acessos paginados é então escolhido.

O segundo passo é repetido tendo-se então o novo agendamento como inicial, até que nenhuma das possíveis subseqüências de alterações melhore o resultado. Depois disso, inicia-se novamente o processo, executando-se o primeiro passo e o segundo passo até que se encontre a melhor combinação de latência e acesso paginado.

A aplicação do método gerou uma melhora de até 18% da letência nos programas testados pelo autor.