"Virtually Pipelined Network Memory"

AGRAWAL, B., Sherwood, T., Virtually Pipelined Network Memory. *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture – MICRO.39,* p. 197-207, December 2006, Orlando, USA

Elaine C. S. Hayashi
RA 068081
Instituto de Computação - Unicamp

1. Resumo*

Primeiramente o artigo apresenta o problema em questão: processamentos em rede executando na sua capacidade máxima demandam não apenas que a memória possa lidar com larguras de banda cada vez maiores (o que significa ter de resolver problemas de conflitos de memória, escalonamento do barramento e outros problemas de *caching*) mas também forneça garantias de rendimento.

A proposta que os autores apresentam para lidar com estes problemas está em controladores que criam a ilusão de uma memória em pipeline (por isso o nome *virtually pipelined memory*), onde as latências dos acessos são padronizadas em um tamanho fixo e previsível e reorganizadas em seqüência para que possam ser processadas em paralelo.

Além da introdução (que é um resumo do artigo) e a parte sobre os trabalhos correlatos existentes, o texto foi dividido em três grandes partes: a seção 3, que apresenta a proposta e explica como ela será sustentada; a seção 4, que detalha a arquitetura; e a seção 5 que traz as provas matemáticas.

Nesta arquitetura proposta, a memória é dividida em vários bancos pequenos, cada um com seu controlador. A figura 3 representa a arquitetura desses controladores e descreve cada um dos seus cinco componentes, cujas principais tarefas são colocar os requerimentos em fila e garantir uma latência fixa. Quando um acesso é feito, a busca é realizada em paralelo e a cada ciclo o dado vai descendo pelo pipeline até chegar ao fim e ser lido pelo processador. Como apenas um acesso está efetivamente acessando o banco por vez, o problema de conflitos está resolvido. Um *stall* só vai ocorrer quando muitos requerimentos chegarem em um curto período de tempo para um mesmo banco e a fila estourar.

Os autores analisam duas razões para um *stall* ocorrer: a primeira, por puro azar, pode ser que mais de 1/B dos requerimentos acabem indo para um único banco (*sendo B o número de bancos). A segunda, ocorre quando acessos redundantes chegam para uma mesma linha de memória (estes casos estão representados na figura 1, nos gráficos do meio e da direita. O gráfico da esqueda, na figura 1 representa uma operação padrão). A solução apresentada para o primeiro caso está nas filas de acesso que vão normalizar a latência para um delay D de 1000 ns (na verdade D vai depender da latência do acesso à um banco e do tamanho das filas). A solução para o segundo caso é detalhada na seção 5.2, quando eles explicam sobre a idéia de fusão de filas (os acessos redundantes seriam agrupados em um único acesso).

Dentro desta arquitetura proposta, são 3 as filas que poderão estourar (3 casos possíveis de *stalls*): a do *Delay Storage Buffer*, do *Bank Request Queue* ou do *Write Buffer*. Boa parte da seção 5 é destinada ao cálculo do tempo médio para ocorrer um *stall* (MTS – *Mean Time to Stall*) para cada uma das 3 filas. Uma estimativa é feita usando números otimais para que se obtenha o melhor MTS possível: um stall a cada 10^{13} acessos à memória e a quantidade ideal de bancos para este caso é 32. Na segunda parte da seção 5, é feita uma generalização do uso da abordagem proposta, mostrando que é possível implementar o mesmo para *Packet Buffering* (podendo suportar um número de filas consideravemente maior, mas com latência menor) e *Packet Reassembly* (para garantir que os pacotes sejam lidos em ordem).

A conclusão, além de repassar cada um dos pontos tratados pelo artigo, menciona também sobre uma futura análise a respeito dos pontos que poderão ser implementados usando a abordagem proposta: inspeção e classificação de pacotes, *application-oriented networking* e até mesmo aplicações mais largas com *streaming* irregular.

2. Conclusão

Os autores conseguiram uma boa solução para o problema do gargalo da memória em processadores de rede, estendendo alguns conceitos já existentes (controladores de memória que têm estágios de pipeline para acessar os dados; bancos de memória; processamento em ordem) para formar a idéia inovadora apresentada neste artigo (e proposta pela primeira vez em 2003). A arquitetura foi exposta de maneira clara e os possíveis problemas foram analisados e comprovados.

*Trabalho proposto pelo Prof. Doutor Paulo Cesar Centoducatte para o curso MO401 Arquitetura de Computadores I