MO401 - Trabalho 1 – Resumo de artigo

Data Access Partitioning for Fine-grain Parallelism on Multicore Architectures
Chu, M., Ravindran, R., and Mahlke, S.; Micro 2007

Resumido por **Flávio Augusto Wada de Oliveira Preto** (RA 032883)

Atualmente, devido as restrições para o aumento do *clock*, os projetistas de hardware estão pendendo para arquiteturas multi-cores. Desta forma, as aplicações precisam se adaptar para poder atingir um alto grau de parelismo e deste forma diminuir o grau de ociosidade do hardware. O artigo resumido busca o aumento do desempenho de aplicações através da melhor exploração de paralelismo em nível de instrução (do inglês, ILP) através do particionamento da aplicação guiado pelo *profile* de acesso a memória.

Partindo de uma arquitetura com dois níveis de cache, sendo o L1 individual para cada core e o L2 comum a todos, além de uma rede de comunicação rápida para escalares, os autores buscam formas de distribuir as instruções de um programa single-thread sem um paralelismo evidente entre os *cores* de forma a minimizar os stalls causado por incoerência de cache.

Com o foco no cache, através de uma análise dinâmica do código objeto, é extraído um grafo de acesso a memória por linha de cache. A partir dos blocos de execução e do grafo de acesso são identificados as interferências entre os blocos (construtivas e destrutivas) e a vida útil deles. Desta forma é possível particionar os blocos, determinando em que core cada instrução deve ser executada, visando a minimizar interferencias destrutivas de caches, reduzindo *stalls*.

Com base nesta proposta, os autores alteraram o compilador Trimaran e executaram os experimentos com programas do *SPEC CPU 2000*, do *Mediabench* e dos kernels de *DSP* em um modelo de arquitetura flexível em tamanho de cache e cores.

Os resultados foram excelentes, com está técnica foi possível melhorar a média de tempo de stall memory em 51%, apesar de alguns casos os ganhos foram superiores a 91%, e um aumento de desempenho da ordem de 30% em média para um processador de 2 núcleos. Para um processador de 4 núcleos os resultados obtídos foram bastante coerentes com os de 2 núcleos, com pequenos desvios devido a natureza dos programas.

Apesar de não ser o objetivo do estudo, foi notado que a redução do tráfego de coerência está intimamente ligada com o aumento do desempenho, já que a redução de tráfego de coerência é efeito direto do particionamento pelo algorítmo que aloja blocos com alta afinidade de memória. Desta forma, os autores especulam que é possível usar o tráfego de coerência como um indicador de performante.

Também foi considerado o impacto no tempo de compilação, que depende essencialmente de três variáveis: o programa de entrada, o tamanho da janela de análise do algorítmo e o número de acessos a memória executada pelo programa. Mesmo considerando que o sistema não é otimizado para uma compilação rápida, o impacto no tempo de compilação foi na ordem de 30%.

Desta forma, o artigo aqui resumido expôs uma técnica para extrair mais desempenho em arquiteturas multi-cores através de uma técnica de particionamento de instruções por afinidade de memória. Esta técnica faz com que instruções mais afins sejam executadas no mesmo core. Portanto, reduzindo o problema de coerência de cache e troca de dados entre cores. Desta forma, o desempenho foi aumentado significativamente, como comprovado pelos resultados dos testes.