Resumo do trabalho

Dataflow Execution of Sequential Imperative Programs on Multicore Architectures

de Gupta e Sohi, University of Wisconsin-Madison, MICRO-44 (2011)
Resumo escrito por Rafael Auler, RA 045840, 31/03/12

Este trabalho propõe um método para execução paralela de programas escritos em linguagens imperativas que não foram criados com paralelismo explícito (chamado de estaticamente paralelo). Portanto, trata-se de uma técnica de extração de paralelismo para um programa sequencial utilizando técnicas baseadas em análise de fluxo de dados. As técnicas tradicionais para extração de paralelismo com análise do fluxo de dados são restritas a programas escritos em linguagens funcionais, mas esta restrição é removida neste trabalho. O autor compara a sua técnica com a extração de paralelismo a nível de instruções: nesta, duas abordagens existem, aquela usada em arquiteturas VLIW, as estáticas, e a análise dinâmica do fluxo de dados usada, por exemplo, no método de Tomasulo. Ele argumenta que a análise dinâmica é mais proveitosa, pois requer compiladores menos elaborados e menos esforço do sistema operacional em resolver problemas que só aparecem em tempo de execução.

A granularidade do paralelismo extraído nesta proposta, entretanto, não é no nível de instruções, em que cada instrução independente executa em uma unidade funcional diferente a fim de aumentar o paralelismo. A ambição é usar informações de dependência de dados para executar funções que usam conjuntos de dados diferentes em diferentes núcleos de processamento (*cores*).

O modelo de execução proposto sugere uma infraestrutura de *hardware* para auxiliar a execução de programas escritos especificamente para este sistema. O usuário deve especificar quais funções são candidatas para serem executadas potencialmente em paralelo e, para tais funções, definir explicitamente seus conjuntos de objetos lidos e escritos. O sistema então implementa aquisição de *tokens* pra cada objeto lido e escrito de forma a serializar a execução em dependências RAW, WAR e WAW nos objetos e executar em paralelo funções independentes nestes quesitos.

Para testar a ideia, um protótipo completamente em software foi construído com o uso de *templates* em C++. O *template* "df_execute", por exemplo, sinaliza que uma função deve ser executada usando a infraestrutura que irá analisar suas dependências e executar outras funções em paralelo. O usuário também especifica barreiras com uma chamada a "df_end". Segundo os autores, impor esta descrição aos usuários é um ônus muito pequeno se comparado com a especificação de paralelização explícita.

Seis programas foram avaliados nesta infraestrutura: barneshut, blackscholes, bzip2, dedup, histogram e reverse_index, comparados contra a implementação usando pthreads (travas tradicionais). A média harmônica do *speedup* alcançado com a paralelização usando a infraestrutura proposta mostra que ela é apenas 13% pior do que o speedup alcançado com a programação paralela explícita utilizando travas e pthreads em um sistema Core i7, 21.7% pior em um AMD 8350 e 18.2% pior em um AMD 8356. Finalmente, os autores concluem argumentando que o futuro da paralelização não está na paralelização estática e explícita, mas em técnicas menos onerosas para os usuários, como a proposta neste trabalho.