Resumo: Idempotent Processor Architecture

DE KRUIJF, M.; SANKARALINGAM, K. Idempotent Processor Architecture. MICRO-44 '11.

Autor do resumo: Rodolfo Guilherme Wottrich – 134077

Este trabalho utiliza o conceito matemático de *idempotência* para propor uma nova arquitetura de processadores, a *Arquitetura Idempotente de Processadores*. Idempotência diz respeito a operações matemáticas que podem ser executadas infinitas vezes e, ainda assim, apresentam o mesmo resultado de que se fossem executadas apenas uma vez (como a multiplicação por 1). Processadores idempotentes executam sequências idempotentes de instruções, permitindo a reprodução precisa do estado do processador através de simples re-execução da região de código idempotente, tornando desnecessário todo o suporte de hardware para recuperação de misspeculation. Isso simplifica o design de processadores in-order e reduz significativamente o consumo de energia, inclusive com ganho de desempenho com relação a processadores in-order comuns.

Dentro de uma região idempotente, tanto a execução quanto o retirement de instruções podem ser realizados out-of-order. Isso permite a utilização imediata de dados produzidos por instruções de baixa latência sem a necessidade de complexos esquemas de staging e bypassing. Além disso, simplifica a implementação de suporte preciso de exceções. Por fim, permite o retirement out-of-order com respeito a operações com latência imprevisível, como loads.

O artigo descreve como identificar e construir regiões de código idempotente. Intuitivamente, uma região idempotente não pode sobrescrever suas entradas, para que possa ser re-executada do início em qualquer momento de sua execução. Isso se traduz, basicamente, em não conter dependências de dados do tipo WAR sem haver antes uma RAW. A esse tipo de dependência, os autores dão o nome de *clobber antidependence*. Há *clobber antidependences* que pertencem à semântica do programa e outras artificiais, geradas por técnicas de register renaming do compilador. Essas últimas podem ser evitadas por compiladores específicos para gerar código com mais idempotência, gerando regiões idempotentes maiores e garantindo melhores resultados com a arquitetura idempotente. Os autores criaram um compilador com essas características usando LLVM e submeteram os códigos a benchmarks SPEC 2006, PARSEC e Parboil e demonstraram que é possível gerar código com muito mais idempotência na maioria dos casos, para arquiteturas ARM e x86.

Processadores idempotentes eliminam a necessidade de muitos elementos do hardware convencional, então os autores ainda propõem a adição de um slice data buffer (SDB) de quatro entradas para uma leve melhora no desempenho. O SDB mantém instruções dependentes de um load miss, desbloqueando o pipeline e permitindo o issue de instruções próximas com respeito às que estão no SDB.

O trabalho ainda realiza uma avaliação experimental da arquitetura, através das mesmas suítes de benchmark, comparando através de simulação combinações de processador in-order comum, processador idempotente simples, processador idempotente com a otimização do SDB, processador out-of-order comum e código original e idempotente. Os resultados demonstram que a arquitetura idempotente otimizada obtém ganho médio de 4.4% no desempenho sobre arquitetura in-order comum, além de trazer redução na complexidade do hardware e redução significativa no consumo de energia (apesar de o artigo não realizar avaliação experimental desse aspecto). Processadores out-of-order ainda executam muito mais eficientemente, mas o objetivo não é exceder a eficiência de arquiteturas desse tipo.

Por fim, os autores ainda discutem alguns aspectos importantes relativos à sua proposta de arquitetura idempotente.

Com este trabalho, conclui-se que repensar alguns dos princípios fundamentais da organização de hardware através de um ponto de vista de restrições de consumo de energia pode melhorar significativamente a eficiência de processadores.