Técnicas de Otimização para Arquiteturas Superescalares

Eduardo D. Camaratta, Tatiana G. S. dos Santos, Philippe O. A. Navaux

Universidade Federal do Rio Grande do Sul – Instituto de Informática
Av. Bento Gonçalves, 9500, Bloco IV, Porto Alegre, Brasil.
{edcamaratta, tatiana, navaux}@inf.ufrgs.br

Introdução

Aplicações cada vez maiores e mais complexas conduzem a indústria de microprocessadores ao rápido desenvolvimento de alternativas para obter ganho de desempenho. Entre essas alternativas é destacável a preferência pelo uso de arquiteturas superescalares pelo seu potencial de execução de várias instruções por ciclo de relógio paralelamente.

Essas arquiteturas possuem alguns problemas instrínsecos de sub-utilização dos recursos, causados sobretudo pelas dependências de dados e instruções. Obter ganho de desempenho através de modificações no hardware tem se tornado um desafio cada vez maior e é notável a busca por novos caminhos. Mecanismos como execução fora de ordem e especulativa, *cache* de traços e execução simultânea de múltiplos *threads* já são realidade em microprocessadores do estado-da-arte, e a adição de novos mecanimos requer um alto esforço de implementação, sem garantias de uma melhora de desempenho significante.

Uma das saídas para se obter o ganho de desempenho desejado é uma utilização dos recursos de forma eficiente. Uma alternativa que vem se destacando em diversas pesquisas tanto na área acadêmica quanto na indústria é a utilização de novas técnicas de compilação. Essas técnicas buscam otimizar o código das aplicações para uma boa utilização das estruturas oferecidas pelas arquiteturas-alvo.

Desse modo, esse trabalho tem como principal objetivo estudar novos mecanismos que podem gerar códigos mais eficientes para as arquiteturas que vem sendo estudadas e desenvolvidas no âmbito do projeto APSE.

A Ferramenta *Simplescalar*

A implementação de uma arquitetura superescalar de propósito geral não é uma tarefa trivial. Tipicamente, a primeira fase do desenvolvimento de novas tecnologias a serem empregadas em novos projetos engloba a implementação de simuladores em linguagens de alto nível. Isso é necessário para que os projetistas tenham uma idéia do desempenho que será alcançado por essas novas tecnologias e a partir daí decidir se a implementação propriamente dita é realmente viável.

Atualmente a ferramenta *SimpleScalar* se destaca por ser um conjunto de simuladores *open source* totalmente parametrizável que fornece várias estatísticas para auxílio na avaliação de desempenho de arquiteturas modernas [AUS 02]. O *SimpleScalar é* composto por cinco simuladores básicos: *sim-fast* (para simulações rápidas e pouco detalhadas); s*im-cache* (para simulação de cache); *sim-cheetah* (para

gerar múltiplos resultados de simulações para diferentes configurações de cache); *sim-profile* (para geração de detalhados perfis de classes de instruções e endereços, acessos a memória e desvios) e; *sim-outorder* (o mais complicado e detalhado simulador, para simulação de execução de instruções fora de ordem).

Juntamente com os simuladores são disponibilizadas aplicações para *Benchmarking* (SPEC95 e SPEC2000), bem como uma versão do Compilador *Gnu C* (ou *GCC SimpleScalar*) necessária para produzir aplicações para o ISA específico do *SimpleScalar*, chamado PISA. Esse conjunto de instruções é similar ao MIPS, com algumas poucas modificações.

Assim como diversos grupos de pesquisa espalhados pelo mundo, os projetos desenvolvidos pelo grupo APSE do Instituto de Informática da UFRGS utilizam os simuladores da ferramenta *SimpleScalar* em seus projetos. Diversas Teses e Dissertações já foram desenvolvidas baseadas, principalmente, no simulador *Sim-Outorder* [SAN 03a], [GON 00], [PIL 04] que oferece a configuração de arquiteturas mais robustas, apresentando resultados mais próximos do real. Os simuladores modificados, no entanto, fornecem arquiteturas com peculiaridades e características diferentes de arquiteturas superescalares convencionais, adequando-se a cada uma das pesquisas desenvolvidas.

A Figura 1 apresenta o fluxo de ferramentas para desenvolvimento de arquiteturas na ferramenta *SimpleScalar*.

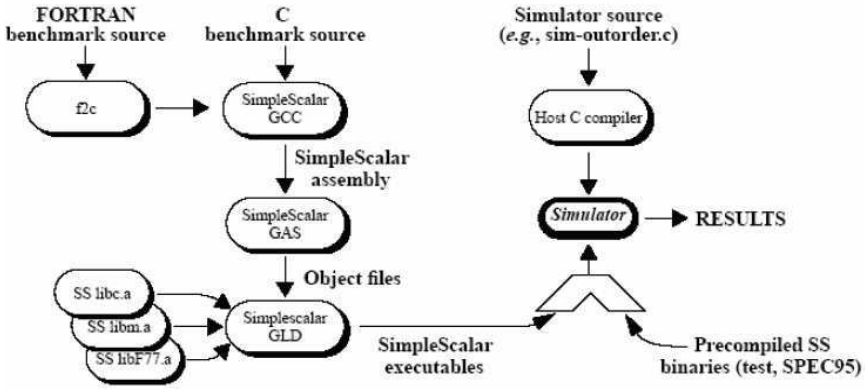

Figura 1: Fluxo de desenvolvimento da ferramenta SimpleScalar

Os fontes dos *benchmarks* são compilados com o *GCC SimpleScalar*. A seguir são utilizados o *assembler* e o *linker* específicos. Os executáveis gerados são então utilizados juntamente com um simulador, seja ele o *sim-outorder* ou qualquer outro simulador da ferramenta. Desse modo, é possível obter resultados detalhados da execução dessa nova aplicação na arquitetura modelada pelo simulador.

Como já discutido, o acréscimo de mecanismos sofisticados a já complexas arquiteturas superescalares não oferece garantia de desempenho ótimo. Uma alternativa é estudar qual o impacto do código compilado com diversas opções de otimização diferentes em uma dada arquitetura. A idéia é identificar em que situações cada uma dessas opções afeta o desempenho. A arquitetura alvo escolhida é a chamada DCE (*Dynamic Conditional Execution architecture*), desenvolvida em uma Tese de Doutorado do grupo APSE [SAN 03a].

Sendo assim, o objeto principal desse estudo é o compilador *GCC Simplescalar*. Em uma estapa futura, será possível a introdução de modificações no próprio

compilador a fim de modificar o código das aplicações com intuito de obter um melhor desempenho na execução na arquitetura DCE.

A Arquitetura DCE

Entre os simuladores desenvolvidos pelo grupo APSE está o s*im-dce* [SAN 03a]. Esse simulador modela a arquitetura DCE ou *Dynamic Conditional Execution architecture*, que é uma arquitetura baseada em execução condicional dinâmica. O *sim-dce* é baseado no simulador *sim-outorder* do *SimpleScalar* e obedece o mesmo fluxo, com as mesmas ferramentas de suporte, que o simulador original. A principal característica dessa arquitetura é um mecanismo dinâmico de predicação de desvios. Com o uso da predicação em alguns casos específicos de desvios, a penalidade produzida por previsões incorretas pode ser reduzida. Isso se deve ao simples fato da arquitetura executar um número menor de previsões. A completa descrição dessa arquitetura pode ser vista em [SAN 03a].

De fato, a arquitetura DCE foi escolhida como arquitetura alvo desse estudo porque o seu compilador já exerce uma grande influência nos resultados da arquitetura. No DCE, o compilador marca os desvios que podem ser predicados (ou qualificados para predicação), seguindo algumas restrições com relação ao tamanho e formato das estruturas aninhadas. Essas estruturas de desvios serão efetivamente predicadas apenas se existem recursos disponíveis no momento da sua execução. A partir daí, os desvios predicados terão seus dois caminhos (tomado e não-tomado) executados e um deles é escolhido para compleção, de acordo com a saída correta do desvio. Os desvios não qualificados para predicação são previstos normalmente, como em uma arquitetura superescalar convencional.

Desse modo, é crucial que haja um grande número de desvios qualificados. Quanto mais desvios forem qualificados, maior é a flexibilidade da arquitetura para escolher dinamicamente quais serão efetivamente predicados.

É, portanto, notável a importância do compilador para um bom aproveitamento dos recursos oferecidos pela arquitetura.

O Trabalho Proposto

Dentro desse contexto, esse trabalho tem como principal objetivo o estudo de novas técnicas de otimização de código para o DCE. A idéia é poder gerar, usando o compilador *Gnu C* para PISA, um código específico para essa arquitetura. Dessa forma, o desempenho da arquitetura deve aumentar significativamente e alguns dos mecanismos hoje implementados diretamente no hardware podem ser otimizados.

Trabalhos anteriores [SAN 03b] já estudaram os conjuntos de opções de compilação mais conhecidas no *Gcc* (*–O1, –O2* e *–O3*), e concluíram que as otimizações incluídas na opção *–O2* são as que produzem um maior número de desvios qualificados.

Desse modo o presente trabalho busca identificar quais das otimizações presentes na opção *–O2*, efetivamente causam tal resultado.

Baseados nesses novos experimentos, técnicas de compilação que reflitam otimizações do mesmo tipo serão estudadas e posteriormente implementadas no compilador *Gcc* da arquitetura. Espera-se que com isso, um conjunto maior de desvios qualificados seja encontrado e a arquitetura possua uma gama maior de estruturas que

possam ser predicadas. Mesmo não sendo garantia para um melhor desempenho, espera-se que o aumento do universo de predicação da arquitetura signifique que os recursos disponíveis sejam ainda melhor aproveitados.

Conclusões e Trabalhos Futuros

O compilador *GCC SimpleScalar* já encontra-se funcional. Também já foram identificadas as *flags* de otimizações específicas acionados por cada opção de compilação. Os próximos passos compreendem a recompilação dos benchmarks SPEC com objetivo de analisar quais das otimizações dentro das opções *–O1, –O2* e *–O3* geram efetivamente códigos com mais desvios qualificados para predicação na arquitetura DCE.

Com essas modificações espera-se que seja possível identificar que tipo de otimizações efetivamente impactuam nos resultados de qualificação de desvios. A partir daí será possível identificar novos modos de otimização e modificar o próprio compilador para refletir tais modificações.

Referências

[AUS 02] AUSTIN, T.; LARSON, E.; ERNST D. **SimpleScalar:** An infrastructure for computer system modeling. *IEEE Computer*, 35(2):59–67, Feb. 2002.

[SAN 03a] SANTOS, R. R. dos. ***DCE:*** *The Dynamic Conditional Execution in a Multipath Control Independent Architecture*. PhD thesis, Instituto de Informática, Universidade Federal do Rio Grande do Sul, May 2003.

[SAN 03b] SANTOS, R. R, SANTOS, T. G. S.; PILLA, M. L.; NAVAUX, P. O. A.; BAMPI S.; NEMIROVSKY, M. Complex Branch Profiling for Dynamic Conditional Execution. In *Proceedings of the SBAC-pad 2003*. São Paulo, 2003. IEEE.

[GON 00] GONÇALVES, R. A. L. ***Simultaneous Multithreaded Architectures: SEMPRE:*** *A SMT Architecture with Capacity of Execution and Scheduling of Processes*. PhD thesis, Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2000.

[PIL 04] PILLA, M. L. ***RST:*** *Reuse Through Speculation on Traces*. Ph.d. thesis, Instituto de Informática, Universidade Federal do Rio Grande do Sul, June 2004.