ESTIMATIVA DE DESEMPENHO DE PROGRAMAS

Diogo Takayuki Murata (PIBIC/CNPq/UEM), Anderson Faustino da Silva (Orientador), e-mail: anderson@din.uem.

Universidade Estadual de Maringá / Centro de Tecnologia /Maringá, PR.

Ciência da Computação, Sistemas de Computação

Palavras-chave: compiladores, otimizações, desempenho.

Resumo:

No contexto de compiladores, o desenvolvimento de seletores de otimizações é um desafio nos dias atuais. Abordagens para a implementação desses seletores são encontradas na literatura e envolvem o uso de buscas aleatórias, exaustivas e heurísticas, algoritmos genéticos e aprendizagem de máquina. Tais trabalhos tem demonstrado que é possível melhorar o desempenho dos programas à medida que a escolha do conjunto de otimizações a serem aplicadas seja feita por um processo guiado. Contudo, tais trabalhos incorrem no problema de necessitarem de um elevado tempo de execução, pelo fato da necessidade de compilar e executar o programa diversas vezes para validar o conjunto de otimizações a ser utilizado. Neste contexto, este artigo descreve um método para estimar o desempenho de programas, quando compilados com determinados conjuntos de otimizações sem a necessidade de sua execução real.

Introdução

Compilador é uma ferramenta que traduz código de linguagens de alto nível, legíveis a seres humanos, em linguagem de montagem (Assembly) (AHO e outros, 2006) (BLUM, 2005). Essa tradução não é trivial e é feita em etapas, que podem variar de acordo com a implementação do compilador.

Atualmente, os compiladores fornecem diversas otimizações (transformações) cujo objetivo é melhorar a qualidade do código final. Dentre as otimizações é possível escolher quais serão aplicadas, pois nem toda otimização modificará o código de maneira benéfica, podendo em alguns casos até ocasionar perda de desempenho.
Neste contexto, o Problema da Seleção de Otimizações (PSO) (de LIMA e outros, 2013) é: escolher o melhor conjunto de otimizações para um código

X, tal que X seja ótimo para um dado objetivo. Uma forma de garantir o resultado ótimo é avaliar todos os possíveis conjuntos de otimizações, levando em consideração a ordem e seus parâmetros. No entanto, não é difícil perceber que tal metodologia é inviável em termos de tempo de processamento. Considerando tão somente os conjuntos de otimizações possíveis, sem considerar a ordem e os parâmetros, a quantidade de avaliações (compilar e executar o programa) para um determinado código seria de 2^n, onde n é o número de otimizações possíveis de serem aplicadas. Assim, um compilador que oferece a possibilidade de aplicação de 61 otimizações diferentes, como é o caso do LLVM (LATTNER, 2002), necessitaria de: $2^n = 2^{61} = 2.305.843.009.213.693.953$ avaliações.

Isso significa que para garantir o melhor subconjunto de otimizações, fazendo a experimentação de todas as combinações possíveis de um programa que compila e executa em 1 segundo, seriam necessários cerca de 73 bilhões de anos de processamento, o que é impraticável.

Na tentativa de procurar soluções viáveis para o PSO, várias abordagens foram propostas na literatura. Dentre essas abordagens estão: buscas exaustivas, técnica estatística, eliminação iterativa, algoritmos genéticos, aprendizagem de máquina e também abordagens que combinam duas ou mais dessas técnicas. Diversas estratégias demonstraram ser eficientes, contudo ao custo de elevado tempo de resposta.

O presente artigo apresenta uma estratégia para estimar o desempenho de programas, quando compilados com determinados conjuntos de otimizações, sem a necessidade real de sua execução. Desta forma, será possível reduzir o tempo de resposta das estratégias de descoberta de conjuntos de otimizações.

Materiais e métodos

Para estimar o desempenho de um programa sem a necessidade real de executá-lo, é necessário estabelecer as características que determinam o tempo de execução de um código. Basicamente, o desempenho de um programa pode ser estimado por suas características estáticas, mais especificamente, pelas características do código Assembly.

Cada instrução Assembly possui um custo determinado, em ciclos de clock, o qual é especificado durante o desenvolvimento do hardware em questão. Desta forma, é possível propor uma estratégia que estime o desempenho do programa levando em consideração o custo de suas instruções Assembly.

O algoritmo proposto para estimar o desempenho do programa, segue os seguintes passos:

1. A partir do código Assembly, criar o grafo de fluxo de controle do código;
2. Analisar cada bloco básico, especificando o seu custo a partir dos ciclos de clock de suas instruções; e
3. Determinar os laços existentes no grafo de fluxo de controle.

Após a execução do algoritmo proposto é possível fornecer uma estimativa do desempenho (ou em outras palavras, do tempo de execução), de um programa específico. Tal desempenho é dado pelo somatório dos custos de cada bloco básico, considerando que o custo dos blocos básicos que compões um laço são multiplicados por um fator.

Uma questão importante, a ser ressaltada no uso da estratégia proposta, é o fato de não ser possível determinar estaticamente, a quantidade exata de execuções de cada laço do código. Desta forma, é utilizado apenas um multiplicador (fator) para cada laço em questão.

Para implementar o algoritmo proposto, foi escolhida a ferramenta chamada MAO (Micro-architectural optimizer) (HUNDT e outros, 2011). Desenvolvida por pesquisadores da Google, esta ferramenta permite fazer análise e otimização de código Assembly. O atrativo desta ferramenta é a sua capacidade de gerar o grafo de fluxo de controle de um programa, e determinar os laços existentes neste grafo.

Desta forma, a implementação do projeto compreende os seguintes passos:

1. Adicionar à ferramenta MAO o custo dos blocos básicos;
2. Adicionar ao algoritmo de detecção de laços da ferramenta, a funcionalidade de fornecer um custo à cada laço detectado; e
3. Ao final, fornecer o custo total do código analisado.

Resultados e Discussão

Um problema técnico encontrado, durante o desenvolvimento do projeto, foi a instabilidade da ferramenta MAO em identificar os laços que compõem o código em análise. Desta forma, na data de submissão deste artigo, a ferramenta ainda não está finalizada. O que comprometeu a apresentação de resultados finais, no presente artigo.

Atualmente estamos trabalhando para solucionar o problema em questão e assim finalizar o projeto.

Conclusões

Com o avanço da tecnologia e a criação de novas estratégias para as arquiteturas de hardwares, se faz necessário a inclusão de novas otimizações, aos compiladores. Consequentemente, aumenta a pressão dos sistemas que tentam mitigar o PSO.

Neste trabalho descrevemos uma estratégia para estimar o tempo de execução de um programa e desta forma reduzir o tempo de execução dos algoritmos, cujo objetivo é mitigar o PSO.

Os próximos passos do trabalho são: (1) finalizar a ferramenta de estimativa e (2) comparar os resultados encontrados com tempos reais.

Agradecimentos

Agradecemos a UEM e ao CNPq, por proporcionarem a execução deste projeto.

Referências

AHO, A. V., LAM, M. S., SETHI, R., and ULLMAN, J. D. Compilers: Principles, Techniques, and Tools, 2 ed. Prentice Hall, Boston, MA, USA, Sept. 2006.

BLUM, R. Professional Assembly Language. Programmer to Programmer. Wiley, Indianapolis, Indiana, USA, 2005.

HUNDT, R. RAMAN, R. THURESSON, M., VACHHARANJANI, N. MAO – An Extensible Micro-Architectural Optimizer. In IEEE/ACM International Symposium on Code Generation and Optimization, 2011.

LATTNER, C. LLVM: An Infrastructure for Multi-Stage Optimization. Master's thesis, Computer Science Department, University of Illinois at Urbana-Champaign, Urbana, IL, Dec. 2002.

De LIMA, E., de SOUZA, T. X., da Silva, A. F., Ruiz, B. Compiling for performance and power efficiency, In Power and Timing Modeling, Optimization and Simulation (PATMOS), 2013, 23rd International Workshop on (2013), pp. 142-149.