Otimização Unroll and Jam em Fortran através de refatoração de código

Cristian Flores Castañeda[1], **Nicolas Maillard**[1]

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brasil

{cffcastaneda, nicolas}@inf.ufrgs.br

1. Introdução

A obtenção de melhores índices de desempenho pode ser alcançada otimizando o programa em nível de compilação [Aho et al. 1986], ou através da refatoração do código fonte, que consiste no melhoramento da estrutura interna do *software* sem afetar o seu comportamento externo. Estas refatorações tendem a dificultar a legibilidade do código, por isso há uma necessidade de ferramentas como as encontradas em [Overbey et al. 2005] que possam automatizar estas reestruturações.

Unroll and Jam é um técnica de otimização de laços aninhados que visa desenrolar múltiplos laços aninhados e fusioná-los de modo a reduzir o número de desvios e operações na memória [Crawford and Wadleigh 2000]. A Fig.1 e Fig.2 mostram respectivamente um laço original em Fortran, e o mesmo otimizado através de *Unroll and Jam*.

```
DO I = 1, N
 DO J = 1, N
  A(I,J) = A(I,J) * B(I,J)
 ENDDO
ENDDO
```

Figura 1. Laço original.

```
DO I = 1, N, 2
 DO J = 1, N
  A(I,J) = A(I,J) * B(I,J)
  A(I+1,J) = A(I+1,J) * B(I+1,J)
 ENDDO
ENDDO
```

Figura 2. Laço refatorado.

Neste trabalho é realizado a criação de um mecanismo de refatoração capaz de otimizar laços aninhados através de *Unroll and Jam*, a implementação é realizada com base na ferramenta de programação e refatoração Photran [1] e a linguagem alvo é Fortran.

2. Implementação

A estrutura fundamental para a reestruturação de código é a *Abstract Syntax Tree* (AST), pois ela fornece uma representação interna do código fonte gerado diretamente pelo parser utilizado. O Photran permite gerar ASTs editáveis que podem ser modificadas pela manipulação de seus nós [Overbey and Johnson 2009]. Outra estrutura importante no desenvolvimento da refatoração no Photran é o *Virtual Program Graph*, que agrega informações extras nas ligações entre os nós da AST.

As funcionalidades que realizam a reordenação dos nós da AST estão contidas na estrutura *org.eclipse.photran.internal.core.refactoring*. Elas percorrem e modificam os nós da AST dos laços contidos na estrutura *org.eclipse.photran.internal.core.analysis.loop*. Esta última estrutura provê a construção e a representação necessária da AST dos laços.

[1]Eclipse platform technical overview. Technical report, The Eclipse Foundation. Beaton, W. and d. Rivieres, J. (2006).

3. Resultados de desempenho

Os resultados de desempenho foram obtidos utilizando uma máquina com processador Core 2 Duo T8100 2.10GHz, 64KB(32KB para dados e para instruções) de cache L1 por core, 3MB de cache L2 compartilhada, 4GB RAM, kernel linux 2.6.32-23, compilador da intel. Na experimentação foram analisadas, além do algoritmo orginal, 8 versões do algoritmo clássico de multiplicação de matrizes(3 laços aninhados) 1000x1000 em Fortran: 4 versões refatoradas e 4 versões otimizadas pelo compilador Intel. O *Unroll factor* X_Y_Z utilizado nos gráficos abaixo, indica fator de desenrolamento X para o laço mais externo, Y para o laço do meio, Z para o laço interno. Cada versão foi executada 30 vezes, e as avaliações foram realizadas atraves do tempo de execução(s), e do número de requisições de leituras na cache L1-I obtido através do Vtune Analyzer.

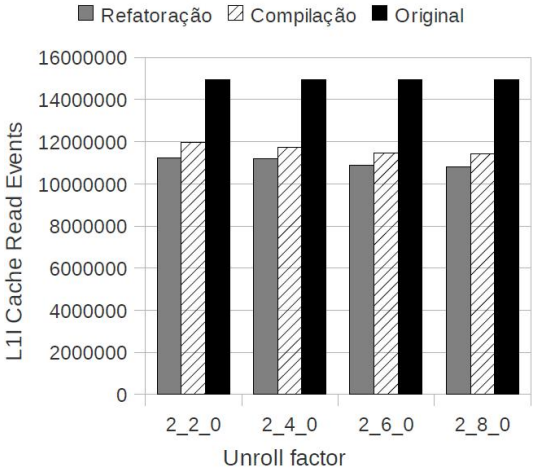

Figura 3. Requisições de leitura na cache L1.

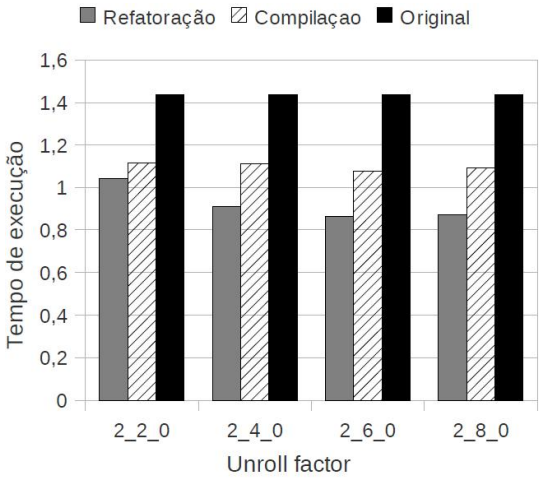

Figura 4. Tempo de execução.

A partir destes resultados na Fig.3 e Fig.4, é possível concluir que os algoritmos refatorados tiveram menos requisições à memória cache L1-I, pois realizaram um melhor aproveitamento na utilização dos registradores do que os algoritmos compilados com diretivas de otimização. Desta forma, é possível considerar que o tempo de execução menor dos algoritmos refatorados são originados pelo melhor aproveitamento da cache L1, pois as requisições de leitura nesta memória foram menores.

Referências

Aho, A. V., Sethi, R., and Ullman, J. D. (1986). *Compilers. Principles, Techniques and Tools*. Addison Wesley.

Crawford, I. L. and Wadleigh, K. R. (2000). *Software Optimization for High Performance Computers*. Prentice Hall PTR, Upper Saddle River, NJ, USA.

Overbey, J., Xanthos, S., Johnson, R., and Foote, B. (2005). Refactorings for fortran and high-performance computing. In *SE-HPCS '05: Proceedings of the second international workshop on Software engineering for high performance computing system a applications*, pages 37–39, New York, NY, USA. ACM.

Overbey, J. L. and Johnson, R. E. (2009). Generating rewritable abstract syntax trees. In *Software Language Engineering: First International Conference, SLE 2008, Toulouse, France, 2008*, pages 114–133. Springer-Verlag.