Refatoração de código Fortran através de Unroll and Jam

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 refatoração consiste na alteração da estrutura interna do *software* para torná-lo mais fácil de ser entendido e menos custoso de ser modificado, sem alterar o seu comportamento observável [Fowler 1999]. Em *High Performance Computing* 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. Estas refatorações tendem a dificultar a legibilidade do código, por isso há uma necessidade de ferramentas como as encontradas em [Overbey et al. 2005] e [Boniati and Charão 2008] que possam automatizar estas reestruturações. Outras aplicações destas ferramentas são utilizadas em trabalhos como em [Pousa and Méhaut 2009], como métodos de pré-processamento para novos módulos.

Unroll and Jam é um técnica de refatoração de laços aninhados que se destaca pela eficiência no uso da hierarquia de memória, através da reordenação da execução dos blocos de código internos dos laços [Crawford and Wadleigh 2000]. Neste trabalho é proposto a implementação de *Unroll and Jam*, e a avaliação do desempenho obtido desta refatoração em relação às aplicadas em nível de compilação.

2. Unroll and Jam

Unroll and Jam visa desenrolar múltiplos laços aninhados e fusioná-los de modo a reduzir o número de desvios e operações na memória. O fator para desenrolar os laços está diretamente ligado ao número de registradores disponíveis [Crawford and Wadleigh 2000]. As Fig.1 e Fig.2 mostram respectivamente a etapa de desenrolar e fusionar dos laços referentes ao código original da Fig.3.

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

Figura 1. Unroll.

```
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. Jam.

3. Implementação

A implementação de *Unroll and Jam* será realizada através das estruturas de dados e funcionalidades da ferramenta Photran [1], e a linguagem alvo será Fortran 77 e 90.

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

A *Abstract Syntax Tree* é a representação interna do código fonte gerada diretamente pelo parser utilizado, sendo fundamental na reestruturação de código. Dessa forma, para possibilitar o processo de refatoração no Photran, o parser foi produzido pelo *Ludwig*, pois ele permite gerar ASTs editáveis que podem ser modificadas pela manipulação de seus nodos [Overbey and Johnson 2009]. Outra estrutura importante no desenvolvimento da refatoração no Photran é o *Virtual Program Graph*. O VPG agrega informações extras nas ligações entre os nodos, no qual junto com a AST fornecem os mecanismos necessários para a refatoração.

As funcionalidades que irão realizar a reordenação dos nodos da AST deverão estar contidas na estrutura *org.eclipse.photran.internal.core.refactoring*. Elas irão percorrer e modificar os nodos 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. Na Fig.3 é possível observar o código dos laços originais em Fortran, e sua AST correspondente na Fig.4. Ambas figuras estão justapostas para facilitar a comparação.

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

Figura 3. Laço original.

Figura 4. AST no Photran.

Referências

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

Boniati, B. B. and Charão, A. S. (2008). Refatoração de programas fortran de alto desempenho. In *Escola Regional de Alto Desempenho*, Santa Cruz do Sul, Brasil.

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

Fowler, M. (1999). *Refactoring: Improving the Design of Existing Code*. Addison-Wesley, Boston, MA, 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.

Pousa, C. R. and Méhaut, J.-F. (2009). Minas: Memory affinity management framework. Technical report, INRIA.