Análise automática de acessos concorrentes a dados para refatoração de código sequencial em código paralelo OpenMP

Dionatan Kitzmann Tietzmann[1], **Andrea Schwertner Charão**[1]

[1]Programa de Pós-Graduação em Informática (PPGI)
Universidade Federal de Santa Maria (UFSM)
Santa Maria – RS – Brazil

{dionatan, andrea}@inf.ufsm.br

1. Introdução

A transformação manual de programas sequenciais em código paralelo requer muito esforço e atenção do programador, que corre grande risco de introduzir erros. Um problema fortemente ligado à programação paralela com memória compartilhada é a condição de corrida, que ocorre em virtude da manipulação concomitante realizada por mais de uma *thread* sobre uma variável compartilhada.

Alguns trabalhos abordam esta problemática, propondo algoritmos e ferramentas que auxiliam o programador durante a paralelização, focando em uma linguagem e/ou ambiente de programação paralela específicos [Basupalli et al. 2011]. O presente trabalho segue nesta linha, propondo a identificação automatizada variáveis que podem vir a ter problemas de condição de corrida. Para tanto, foi projetado e implementado um algoritmo de verificação baseado no acesso às variáveis em programas Fortran. As seções seguintes apresentam uma visão geral sobre o algoritmo, juntamente com alguns resultados obtidos. Mais detalhes encontram-se na dissertação original [Tietzmann 2011].

2. Algoritmo de Análise

O algoritmo de análise recebe como entrada um trecho de código que o programador deseja paralelizar. Com base neste trecho, são analisados os acessos às variáveis na busca por situações que podem vir a caracterizar um problema de corrida, caso o trecho de código seja executado por várias threads. Após esta análise, o algoritmo terá como saída as variáveis críticas, ou seja, as variáveis que poderão estar envolvidas em condições de corrida. Além disso, também terá como saída a localização destas variáveis no código, com a finalidade de facilitar a visualização dos resultados por parte do programador.

O funcionamento geral do algoritmo é o seguinte: dado um trecho de código de entrada, verifica-se cada uma das variáveis nele contidas, cujo valor esteja sendo alterado. Além disso, é verificado se a mesma variável está sendo lida. A verificação de leitura consiste basicamente em procurar em outras partes do trecho e/ou em procedimentos referenciados pelo mesmo, se a variável em análise está sendo utilizada em alguma operação de leitura além da sua atribuição. Nesta situação, além de seu valor estar sendo escrito, ele também estará sendo lido durante a execução deste trecho de código, sendo este fato um indício de que ela pode vir a ser envolvida em uma condição de corrida durante a execução paralela do trecho. Para possibilitar a verificação, é necessário um analisador sintático que possibilite ao algoritmo obter e atuar sobre a árvore sintática do código.

O algoritmo foi implementado tendo como alvo programas em linguagem Fortran. Para isso, utilizou-se a ferramenta Photran, que possui um analisador sintático para

a linguagem em questão e possibilita a instanciação do algoritmo como uma etapa inicial de uma refatoração para código Fortran. A funcionalidade implementada integra-se ao IDE Eclipse (menu refatoração) sob o nome *"Detect Possible Critical Sections"*. Aproveitando o *framework* de refatoração, implementou-se um recurso que sugere diretivas OpenMP para proteção das variáveis envolvidas em possíveis condições de corrida. Este recurso ilustra como o algoritmo pode servir de base para outros trabalhos.

3. Resultados

A figura 1 apresenta um exemplo de uso da ferramenta, onde visualiza-se o trecho de código analisado, as variáveis identificadas e as diretivas OpenMP sugeridas.

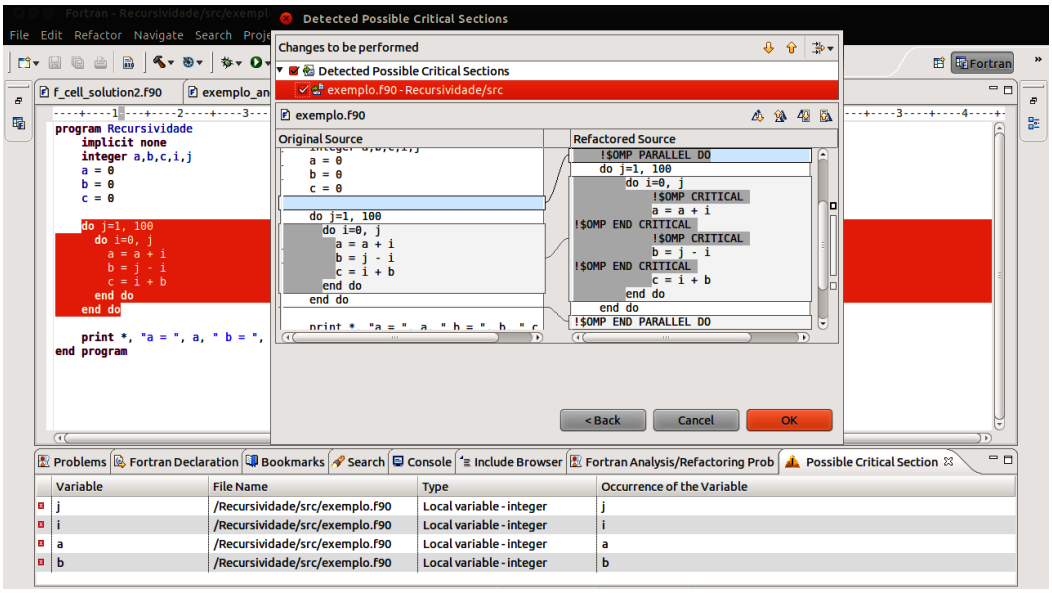

Figura 1. Exemplo de uso da ferramenta

Para fins de avaliação empírica do algoritmo, foram realizados testes com pequenos programas e exemplos de código, mostrando o funcionamento da ferramenta nos casos previstos. Além disso, realizou-se um estudo de caso baseado em uma aplicação real e complexa, mostrando a habilidade do algoritmo em identificar todas as variáveis em risco, bem como ilustrando algumas de suas limitações. De modo geral, os testes apresentaram resultados positivos.

Referências

Basupalli, V., Yuki, T., Rajopadhye, S., Morvan, A., Derrien, S., Quinton, P., and Wonnacott, D. (2011). ompVerify: Polyhedral analysis for the OpenMP programmer. In Chapman, B., Gropp, W., Kumaran, K., and Müller, M., editors, *OpenMP in the Petascale Era*, volume 6665 of *Lecture Notes in Computer Science*, pages 37–53. Springer Berlin / Heidelberg.

Tietzmann, D. K. (2011). Análise automática de acessos concorrentes a dados para refatoração de código sequencial em código paralelo OpenMP. Master's thesis, Universidade Federal de Santa Maria, Santa Maria, RS, Brazil.