Evento	Salão UFRGS 2018: SIC - XXX SALÃO DE INICIAÇÃO CIENTÍFICA DA UFRGS
Ano	2018
Local	Campus do Vale - UFRGS
Título	Paralelização Automática de Linguagens Funcionais baseada no Acelerador ACQuA
Autor	GABRIEL TADIELLO MORAES
Orientador	GABRIEL LUCA NAZAR

Paralelização Automática de Linguagens Funcionais baseada no Acelerador ACQuA

Gabriel Tadiello Moraes – Cartão UFRGS Nº 00277942
Orientador: Prof. Dr. Gabriel Luca Nazar
Universidade Federal do Rio Grande do Sul

Uma forma de se obter um desempenho melhor em um programa é através da exploração dos múltiplos núcleos presentes em quase todos os processadores de propósito geral da atualidade. Para atingir tal objetivo é necessário identificar atividades distintas que podem ser executadas concorrentemente. Essa tarefa nem sempre é trivial, pois é necessário que o desenvolvedor de um programa paralelo leve em consideração questões de baixo nível, como mecanismos de sincronização entre threads e seções críticas. Além disso, de uma forma geral, os programadores estão acostumados a desenvolver algoritmos que sejam executados sequencialmente, pois é uma forma mais intuitiva de se resolver um problema.

Uma ideia que se pode ter frente a esse problema é o desenvolvimento de um algoritmo que automaticamente explore o paralelismo existente em outro programa. Isso também não é tão simples de ser feito: em linguagens procedurais, por exemplo, a ordem de execução das funções é, muitas vezes, importante para o correto funcionamento do programa. Além disso, a existência de porções do código que podem ser executadas de forma simultânea também depende de como esse programa é descrito.

O objetivo deste trabalho é justamente a paralelização automática de códigos escritos em linguagens funcionais. Escolheu-se este conjunto específico de linguagens por causa de uma importante característica das mesmas: a ausência de efeitos colaterais. Por causa disso, a ordem de execução de chamadas de funções independentes não é relevante. Basicamente isso é explorado da seguinte forma: quando uma thread encontra uma chamada de função, ao invés de executar a mesma, esta é colocada em uma fila de funções a serem executadas. Antes de retornar o resultado da função que estava sendo executada, espera-se o resultado das novas chamadas de função. Enquanto elas não retornam, a thread retira uma nova chamada de função da fila. Como qualquer núcleo pode retirar uma nova chamada de função, o programa está sendo executado em paralelo.

Como linguagem de programação para implementação do algoritmo em questão foi escolhido C++ com a biblioteca OpenMP. Como exemplo de linguagem de programação para exploração de paralelismo foi escolhido Haskell. Até o momento, apenas foi realizada tradução manual de algoritmos tradicionais implementados em Haskell para C++ com OpenMP, como forma de verificar a viabilidade e desempenho obtido ao realizar essa tradução. Os resultados obtidos até o momento são promissores: obteve-se um speedup próximo de 4.0 com o algoritmo recursivo de Fibonacci e um speedup próximo de 1.3 com o Merge Sort. Ainda é necessário avaliar essa tradução para outros casos, e por fim desenvolver uma forma de realizar essa tradução automaticamente.